SIMULATION OF DECISION SYSTEMS

C. M. Festa H. W. Adams

MITRE

January 1962

SIMULATION OF DECISION SYSTEMS, Papers and Notes

Taken from a seminar sponsored by the System Sciences Department of The MITRE Corporation

Chairman: H. W. Adams

Editors: Carla-Mae Festa and H. W. Adams

January 1962

THE MITRE CORPORATION

BEDFORD, MASSACHUSETTS

(15) Air Force Contract AF33(600)39852

FOREWORD

On June 6th, 7th, and 8th, 1961 the System Sciences Department of The MITRE Corporation sponsored the first seminar on "Simulation of Decision Systems." The seminar was directed towards the development of general-purpose simulation models of decision-making systems of the class represented by military command and control systems.

Four papers were presented, each of which was followed by an informal discussion among the participants about the paper and allied matters. This report includes the papers and those comments we felt were most pertinent.

The participants were from MITRE and from other organizations who had interest and/or experience with simulation techniques and command and control systems. The backgrounds of the participants were varied; some were trained in psychology, some in engineering, others in mathematics, and still others in fields as varied as anthropology and history.

The first paper was given by Geoffrey Gordon of the IBM Advanced Systems Development Division in White Plains, New York. His paper discusses the generalized modeling technique he and others have developed for the simulation of traffic-handling systems. This technique represents a major advance in efficient handling and representation of system interactions and is, perhaps, the most highly developed technique available.

Next, Irving Wallach of the System Development Corporation group at Paramus, New Jersey, discussed the functional model of a specific command and control system, the Air Force Strategic Air Command Control System (SACCS). This paper indicates the problems associated with modeling complex systems as well as an appreciation of the complexity which characterizes the decision systems with which we are dealing.

In the next two papers Harold Adams, Charles Morrill, and Peter Ten Eyck of The MITRE Corporation present work done on the development of generalized modeling techniques applicable to the design of complex military decision systems. The theoretical approach and preliminary stages of the modeling technique are given together with a specific application of this approach to a problem which occurs in an Air Force command and control system. Essentially, the attempt is made to apply the type of technique presented by Geoffrey Gordon to the problems of modeling systems similar to the system discussed by Irving Wallach.

Editors: Carla-Mae Festa
Harold W. Adams

PARTICIPANTS

Robert L. Barringer	Operations Research Group Arthur D. Little, Inc. 35 Acorn Park Cambridge 40, Massachusetts
Lee S. Christie	Advance Study Project System Development Corporation 2500 Colorado Avenue Santa Monica, California
Paul Coggins	Arthur D. Little, Inc. 35 Acorn Park Cambridge 40, Massachusetts
Geoffrey Gordon	Advanced Systems Development Division IBM Corporation 2 William Street White Plains, New York
Paul Hildebrandt	System Development Corporation The MITRE Corporation P. O. Box 208 Bedford, Massachusetts
William M. Jones, Col., USAF	Joint War Games Control Group Joint Chiefs of Staff Pentagon, Room, 2B 867 Washington 25, D. C.
Peter Kuegel	Arthur D. Little, Inc. 35 Acorn Park Cambridge 40, Massachusetts
Andrew Molnar	Special Operations Research Office 4501 Massachusetts Avenue, N. W. Washington 16, D. C.
Sidney C. Rome	Mathematics & Operations Research Staff System Development Corporation 2500 Colorado Avenue Santa Monica, California

Warren S. Torgerson	Massachusetts Institute of Technology Lincoln Laboratory P. O. Box 73 Lexington, Massachusetts
Irving A. Wallach	Operations Design Branch System Development Corporation 567 Winters Avenue Paramus, New Jersey
Harry Wolfe	Arthur D. Little, Inc. 35 Acorn Park Cambridge 40, Massachusetts

MITRE REPRESENTATIVES

Andrew C. Bayle	V. P. Technical Operation's Office
Harold W. Adams	System Sciences Department
James H. Burrows	Computer Applications Department
Jack Dominitz	SAGE Design & Testing Department
Carla-Mae Festa	System Sciences Department
June Karlson	Advanced Planning Department
Charles S. Morrill	System Sciences Department
Charles V. Riche, Jr.	Data Processing Development Department
Peter H. Ten Eyck	System Sciences Department
David F. Votaw, Jr.	System Sciences Department

CONTENTS

I

AN INTRODUCTION TO GENERAL PURPOSE SYSTEM SIMULATION PROGRAMS

Geoffrey Gordon
IBM Corporation, White Plains, New York

As system designs have become ever more complex, designers have been placing increasing emphasis on the use of digital computers to simulate systems. This development has brought with it the establishment of system study groups containing both engineers and programmers; an enforced union between people with different backgrounds and skills that has not always been happy.

Two major difficulties can arise from the differences in the requirements of engineers and programmers in carrying out their jobs. First, the pace at which they can proceed differs greatly, particularly in the early stages of a project. Engineers can usually produce a large number of possible designs or modifications that may meet the requirements of the system. Writing programs, however, takes time and it is at precisely those places of a project where designs are changing rapidly and the engineer needs the most help from simulation that the programmers are unable to keep up with the pace.

Second, even when the main outline of a system has been established, there is difficulty in getting sufficient flexibility in a simulation program. Initially at least, many details of a system are vague and may be unimportant to the engineer; but a program must be complete and precise before it will run and the programmer needs a complete specification of these details. It is virtually impossible to see beforehand all the possible changes of detail that may occur, and even if it were possible, it would be very difficult to write a program in such a way that it could make allowances for such changes. While the need to change various parameters of the system can be foreseen and allowed for, possible changes in the organization of a system are hard to foresee. Frequently what seem to the engineer to be minor changes in

the structure of a system turn out to require major reorganizations of a simulation program.

System study work is essentially a reiterative process. The performance of a proposed design is compared with the system requirements; the difference is used to propose a new design, and this process is repeated until a successful design is reached. Time taken in changing simulation programs to follow design changes slows down the progress of a project or, more often, results in decisions being made without the benefit of simulation.

These various difficulties are not helped by the fact that engineers and programmers speak different languages. When the actual system design and the simulation model differ, there will be many times when a satisfactory compromise that would minimize the changes needed in a program is overlooked because no one person clearly understands the significant factors of both the design and the program. One way of overcoming these difficulties is to make use of general purpose simulation programs.

The process of producing a simulation study for a digital computer involves two major phases. First, a model of the system to be studied must be constructed and then a program that embodies the logic and action of the model must be written.

The model is a set of equations and logical statements that expresses the relationship between the components of the system and the various parameters associated with the system. The form the model takes and the mode of expression used to describe it depend very much upon the individual engineer setting up the model. Similarly, the form taken by the program written to implement the model depends upon the individual programmer.

Conceptually, the procedure for producing a simulation program falls into two clear parts. In practice, it usually involves a great deal of interaction between the engineer and programmer, thus introducing the possibility of misunderstandings and misconceptions that can lead to difficulties.

If some formal manner of describing the model can be established, then it is possible that the process of putting together a program can be made more automatic and consequently less time consuming. This approach implies that some formal language for describing models can be established which will serve the purposes of both the engineer and the programmer. With such a language, a fair division of labor can be brought about, with the engineer being responsible for finding a satisfactory description of systems within the framework of the special language and the programmer being responsible for producing a program that operates with the language.

The program must be able to deal with any logical statement that may be made in the language, so that it can be used to simulate any system that can be described within this language. To write such a program is a much more severe task to place upon programmers than would be the writing of a simulation program for an individual system. However, the same program would be applicable to all systems that can be described in the program language.

Such a program, therefore, will be general-purpose and it is in this sense that the term general-purpose is used. A general-purpose system simulation program can be defined as a program that is capable of automatically producing a representation of a system from a description of the system expressed in terms of a fixed well-defined system language associated with the program.

In considering what type of language should form the basis of a general-purpose program, the use of block diagrams immediately comes to mind. The majority of people asked to describe a system will resort to a diagram. It is a mode of description that is mutually understood by the many people with different backgrounds who must be familiar with the system. There are many names for such diagrams that describe systems, block diagrams, flow charts, sequence diagrams, node diagrams directed graphs; but the essential meaning is the same.

A block diagram consists of a set of boxes drawn together by lines; the individual actions of the system are described in the boxes and the lines joining the boxes give you the sequence of events going from step to step. There may be

points in the block diagram where there are decisions to be made in order to determine the alternative lines of development.

Although the structure of block diagrams is fairly formal and well understood, the manner in which the actions of the system are described within the boxes varies greatly from system to system and from person to person. The precise way people describe what's going on inside the boxes is very flexible. Each person has his own way of describing these things. Thus, at first sight, there appears to be little formality in the manner of describing systems and consequently there is little hope of finding a well-defined language. However, on a closer look, many common actions can be identified, at least within the area of each specific class of systems.

It is apparent that block diagrams do provide an adequate language for engineering descriptions of systems. To make a language suitable for programming purposes, the manner in which block diagrams are constructed and described must be defined carefully, so that the meaning of each block and its relationship to other blocks is unambiguous. The requirements are, therefore, to provide a well-defined set of basic block types, each with a specific meaning, and to determine a set of conventions controlling the structure of block diagrams.

Many questions arise when considering the possibilities and merits of such an approach to system simulation. The first obvious question is whether, in fact, a suitable language can be found; that is, can you find enough basic block types to describe the actions of a system? If you can, how do you go about it? How do you locate this?

Another big question is: How wide a class of systems can a technique like this be expected to cover? You might find that you can achieve this objective within one particular type of system but once you move from that type of system this language cannot be expanded to cover another area of systems.

Then there is always the question of whether or not you can include an adequate amount of detail in any one system simulation with this approach. Also, will the program be ample to reduce significantly the programming effort required for an individual simulation?

There are many such questions, and inevitably, in designing a general-purpose program, many compromises must be made which will emphasize one aspect of the program's usefulness at the expense of another.

In a block diagram description of a system the basic concept is of some unit of data moving from block to block at certain points in time. The representation of this unit of data in the simulation will be referred to as a transaction. The interpretations to be placed upon the transaction in any particular simulation depends upon the system being simulated. It might be a message in a communication system, a human being or a vehicle in a traffic study, a pulse in a circuit or any one of many other possibilities. The precise interpretation to be placed upon a transaction depends upon the system you want to simulate.

We want to formulate requirements to determine a set of block types each of which has some well-defined property and to establish some formal mechanism for describing interconnections between these boxes such that we can set up and describe the structure of a flow chart. Thus, we hope to be able to describe systems in this language. With this description we should be able to produce a program that will accept any logical statement in the flow diagram or block diagram language and assemble a model automatically.

Given rules for creating transactions and given a description of a block diagram, the action of the simulation program will be to move the transactions through the block diagram step-by-step much in the manner in which a "throw-down" study would be conducted. A "throw-down" study is a term that's used in the communication field for doing simulations by hand; for example, the writing of information on cards and shuffling the cards around to follow the successive steps of the system. This is a technique that has been used very successfully in the study of switching systems.

So our program will operate by moving transactions from block to block following out their progress. For the program to carry out this function when there are many transactions in the system at one time, the simulation program must keep a record of the locations of each transaction to know where a transaction

is at any point in time. It also has to keep a record of what time it is due to move from one block to another block.

Certain types of blocks will describe some interaction between the transactions and an item of equipment associated with the system. It is necessary, therefore, to introduce the concept of an item of equipment as being some element of the system which is engaged or occupied by the transactions during the course of their progress. Again we run into difficulties in that what you define as an item of equipment depends entirely upon the system. However, there are two important factors that will describe almost any item of equipment. First of all, an item of equipment is usually limited in the number of transactions it can handle simultaneously. Secondly, regardless of the action taken by the equipment, it takes a certain length of time to perform this action. A substantial part of any system study is concerned with analyzing the influence of these factors upon the performance of the system. The fact that there is a limited number of transactions that can be handled simultaneously and that it takes a certain length of time to do the equipment's job, results in bottlenecks. A great deal of system study is a matter of balancing the various bottlenecks that exist around the system.

Thus, items of equipment are defined as entities which are characterized by having a capacity, a limit on the number of transactions that can be engaged by the equipment at any one time, and by the fact that they take a certain length of time to carry out their actions. Therefore, the program will have associated with it, items of equipment which will be identified by numbering. Within a given limit the program will allow up to a given number of items of equipment. It may prove to be convenient for programming purposes to distinguish between the type of equipment that can handle only one transaction at a time and the type of equipment that can have multiple occupancy.

The relationship between the item of equipment of the simulation program and the components of the system being studied varies from system to system, and indeed, may vary within one system. For example, a computer in one system might be regarded as a single item of equipment. In another system, the major components

of the computer such as the memory, the arithmetic unit, the various buffers, the input-output channels, etc., might be defined as separate items of equipment each with its characteristic capacity. From this viewpoint the computer is represented as a conglomeration of items of equipment. Another example is a communication network with a trunk system connecting two points which may be regarded as a single item of equipment with a multiple capacity equal to the number of lines. Suppose there were ten lines in this trunk group, then this would be an item of equipment with a capacity of ten. From another point of view your study might be interested in the individual histories of these lines, in which case you will define this trunk group not as one unit but as ten individual items of equipment, each with a capacity of one. One of the advantages of this type of general-purpose program is the greater degree of flexibility.

Some block types will be concerned with engaging equipment during the period of time in which a transaction occupies the block. Some block types will be simple in that they result in a transaction engaging an item of equipment during the length of time that the transaction is in that block. But often the action of the system will be too complex to express, simply in terms of transactions occupying items of equipment for a period of time. There may be many possibilities for the sequence of events to be followed once a transaction has gained control of an item of equipment. The actions of engaging an item of equipment and relinquishing its use should be embodied in separate block types so that, in terms of block diagrams, any amount of complexity can be introduced between the point or points at which a transaction engages and releases the equipment.

Certain common elements can be identified with all block types whether they are concerned with equipment or not. The two most important common elements are the fact that each action represented by a block takes time and each block (with the exception of terminating blocks) has a successor or successors. With these common elements we should have some common mechanism for identifying time and associating successors with each block type.

So, first of all, with every block type there will be associated a block time which will represent the length of time the transaction will spend in that particular block. It obviously would destroy the general-purpose nature of a program to fix the unit of time. Therefore, time is best expressed in terms of integers with the interpretation of the unit integer to be left to the user of the program. This basic unit might very well vary from a microsecond in one system to a day in another system.

The block times will not always be well-known. The time taken for the action described by the block may vary in some arbitrary manner or too little information may be available to define the time accurately. The best thing to do is to arrange that this time can vary over some range which represents your best estimate of the range of values of time. Sometimes you will know very specifically what the time is, but usually you will have only an estimate of some type of mean spread. The simplest thing to do then, is to associate with every block a mean (M_i) and a spread (S_i) such that when any transaction enters the block, you can choose a block time for the particular transaction which lies in the range of mean-minus-spread ($M_i - S_i$) to mean-plus-spread ($M_i + S_i$). You can simply use a random number in this simple square law distribution. There will, however, be occasions when more accurate information about the distribution is known and provision should be made for inserting such known distributions.

If the time for a block is well-known and precise, you obviously can set the spread to zero ($S_i = 0$) so that every transaction has exactly the same time. There will be times when you will want zero time which can be done by setting both the mean and the spread equal to zero ($M_i = S_i = 0$). You have zero time for those blocks that are used as buffers or used as decision points to split the traffic in two or more directions.

Each block may be a successor to many different blocks and conversely, there may be more than one successor to a particular block. Some conventions on the number of possibilities in each case need to be established. The natural way of specifying the links between blocks is by naming at each block the successors to that block. From the point of view of programming, no difficulty is then presented

by the multiplicity of possible inputs to a block but it is very convenient to put an upper limit on the number of possible exits from a block. A limit of two is adequate, since, by cascading with blocks that have zero block time, it is always possible to produce any number of possible paths emanating from a single block.

When there is more than one exit, some mechanism must be established for determining which path will be taken. Sometimes the selection is a random function, the cascade process where the system sends traffic down one or two routes. You do not know where any one specific transaction is going, but you do know that a certain percent of the traffic goes one way and a certain percent goes the other. Each individual action may be chosen randomly. Obviously the program can follow this. For example, you can arrange that you choose between the two alternatives on the basis of a random number weighted by a probability. You specify the probability of going down route one and of going down route two. The program then computes a random number for you, weights it with these probabilities, and selects for you. This type of selection is quite common in systems and can be widely used in the program.

A second important mechanism for choice is the provision of alternatives. There may be a principal exit by which a transaction should proceed but in the event of this path being blocked, because some item of equipment is fully engaged, the transaction proceeds by way of a second exit.

As an example of how such programs can be applied, consider a program based on six basic block types. These different block types are illustrated in Figure 1. Type 1 is represented as an originate block and is a block that creates transactions and passes them on. For this type of block, time will not represent the length of time the transaction stays at that box but rather will represent interarrival time or intercreation time, the interval time between successive creations of transactions.

Type 2 does not involve any equipment; it merely advances the transaction. It accepts transactions, holds them for a certain length of time, and then passes them on. This is going to be useful as a buffer more than anything else.

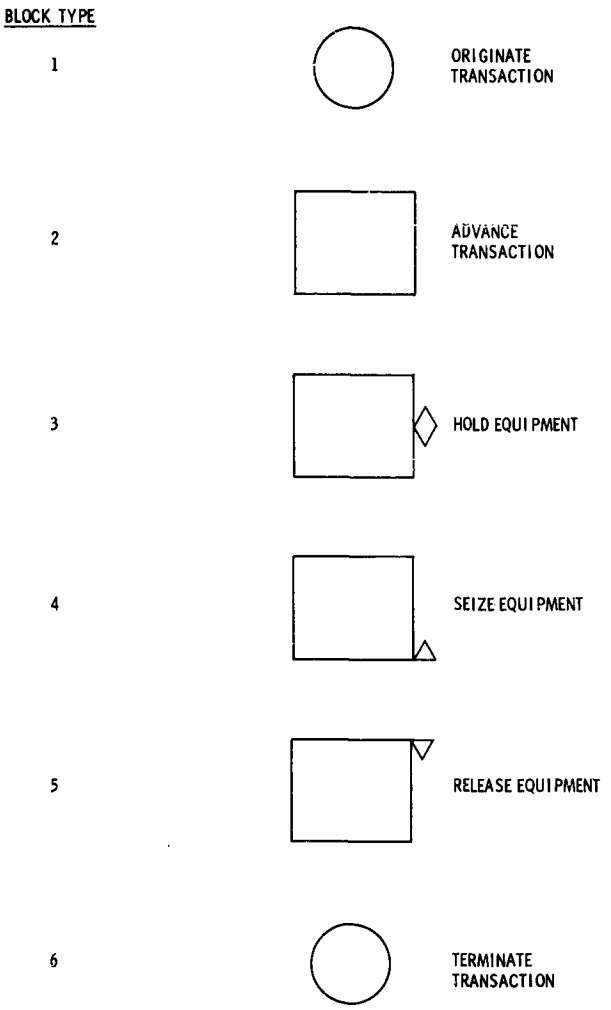


Figure 1. Block Types

Type 3 is known as a hold block. A transaction engages an item of equipment as long as it is in the block. If the equipment is a multiple occupancy type, there may be any number of transactions up to the limit specified.

Type 4 is the seize block in which a transaction engages an item of equipment upon entering the block but does not relinquish the equipment upon leaving. This task is accomplished by Type 5, the release block which effects the release of an item of equipment that has been engaged. As mentioned earlier, the separation of these functions allows you to put any amount of flow diagraming representing the complexity of the system between these two points.

Finally, Type 6 is a terminate block which destroys the transaction after first taking out certain statistics about it. In the symbolic representation, as shown in Figure 1 of these blocks, any item of equipment that is involved is noted by the number of the equipment being placed in a flag attached to the block.

This is a very simple set of block types. The actual written program involves a great deal more. But even with this simple set, I would like to show you how you can assemble a system in the hope that this will illustrate the advantages of this type of approach and what you can expect to get out of it.

As an example of the application of a program based on these blocks, consider the following system. A computer receives a series of input messages and begins processing them. For some messages there is no further action when this processing is complete. For some others, a record must be written on a disc file to complete processing. A third group of messages requires a record to be read from the file before the processing is completed and a fourth group requires both the reading of a record and then the writing of a record. Access to the records in the disc file is gained by way of one of a number of arms. This is a fairly standard type of data processing application.

The manner in which the system can be represented with the block types that have been described is illustrated in Figure 2. In this diagram a transaction is identified as being a message, and two items of equipment are defined: one is the

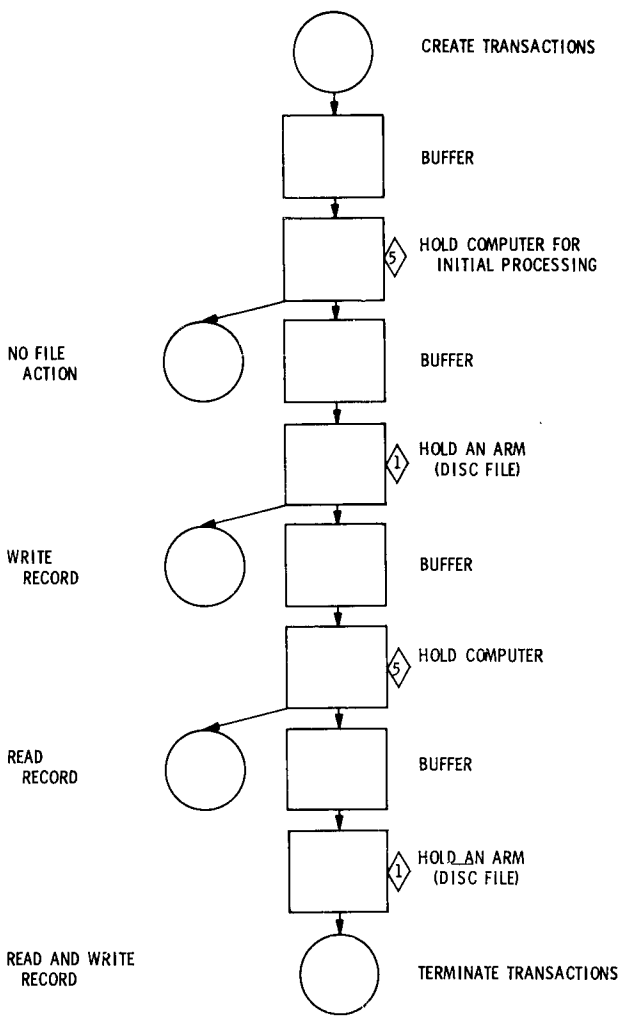


Figure 2. Simple Data Processing System

computer and the other is the disc file. Suppose the computer is only capable of operating on one transaction at a time whereas the disc file has three independent arms each of which, for simplicity, will be assumed to have access to all parts of the disc file without mutual interference. The capacity limits of the items of equipment will be 1 and 3 respectively.

First of all the diagram shows transactions being created in an originate block and then passed into a buffer to await initial processing by the computer. In order to accomplish the initial processing, the transactions must hold the computer for a certain length of time. Those transactions that need no further processing will now go to a terminate block representing those messages that have no file action. The remaining messages will go to a buffer to await the availability of an arm to locate a record position on a disc. These messages are those falling within one of the three classes mentioned: the class in which you have to write a record, the class in which you have to read a record, and the class in which you have to both read and write a record. Some of the messages need access to the disc file to read and some need access to write. Thus, each of these has to hold the disc file.

In this example there are two pieces of equipment and the flags attached to the blocks will identify which piece of equipment is concerned. The computer item will be numbered as equipment number 5 and the disc file as equipment number 1. After an arm is located, those transactions, representing records that only require writing to complete the processing, will then go to the terminate block. The remainder of the transactions return to the computer for processing. Those transactions that only need a record to be read before completion of their processing will then go to the terminate block, while the remaining messages will repeat the process of engaging an arm to write back a record.

This is a very simple representation of the processing system. I have not put in all the details because this is only the structure of the flow chart. Times have yet to be filled in the blocks. You can estimate the various times best by finding out how long you expect this processing to be and how long it takes, on the average, to gain access to the disc file. This demonstrates the point of using spreads. For example,

access to this file is not a constant time but depends upon the location of the discs. Thus it is convenient to represent this time by a mean and a spread. The blocks acting as buffers will have zero time since their purpose is simply to hold transactions which are held up because the equipment is busy. The probabilities with which transactions go down the various paths control the mixture of message types.

This representation will provide information about the general flow of data and about the occupancy of the equipment. The effects of various computing speeds or channel speeds can be studied by means of changing block times. You can study the effect of different workloads by choosing the probabilities of going down these paths to represent different mixes of types of messages, or you can change the equipment capacities to study the effect of processing transactions in parallel or the effect of the capacity of the disc file.

Suppose it is desired to represent the disc file more fully. Instead of treating the file as a single piece of equipment, let each arm be defined as a separate piece of equipment with a capacity of one. Suppose, also, that the input-output channel between the computer and disc file is introduced as a further piece of equipment. It is assumed initially to have a capacity of one and the same channel is used for both input and output.

The system that will be assumed now is one in which access to the disc requires that, first, an arm should be engaged and positioned, the record should then be found, the channel must be engaged to transmit the record, and then the use of the arm is relinquished for the next transaction.

Assuming the traffic is divided evenly between the arms, the section of the block diagram concerned with the access to the disc file can be represented as shown in Figure 3. Here the traffic is shown as being divided between three arms but the details of only one arm are shown. The same item of equipment corresponding to the channel appears twice, once for the input and once for output to the disc file. This same item will also appear in the same way in each line because all arms share the channel. In addition, the item of equipment corresponding to the arm appears twice in each line, once for a writing access and once for a reading access.

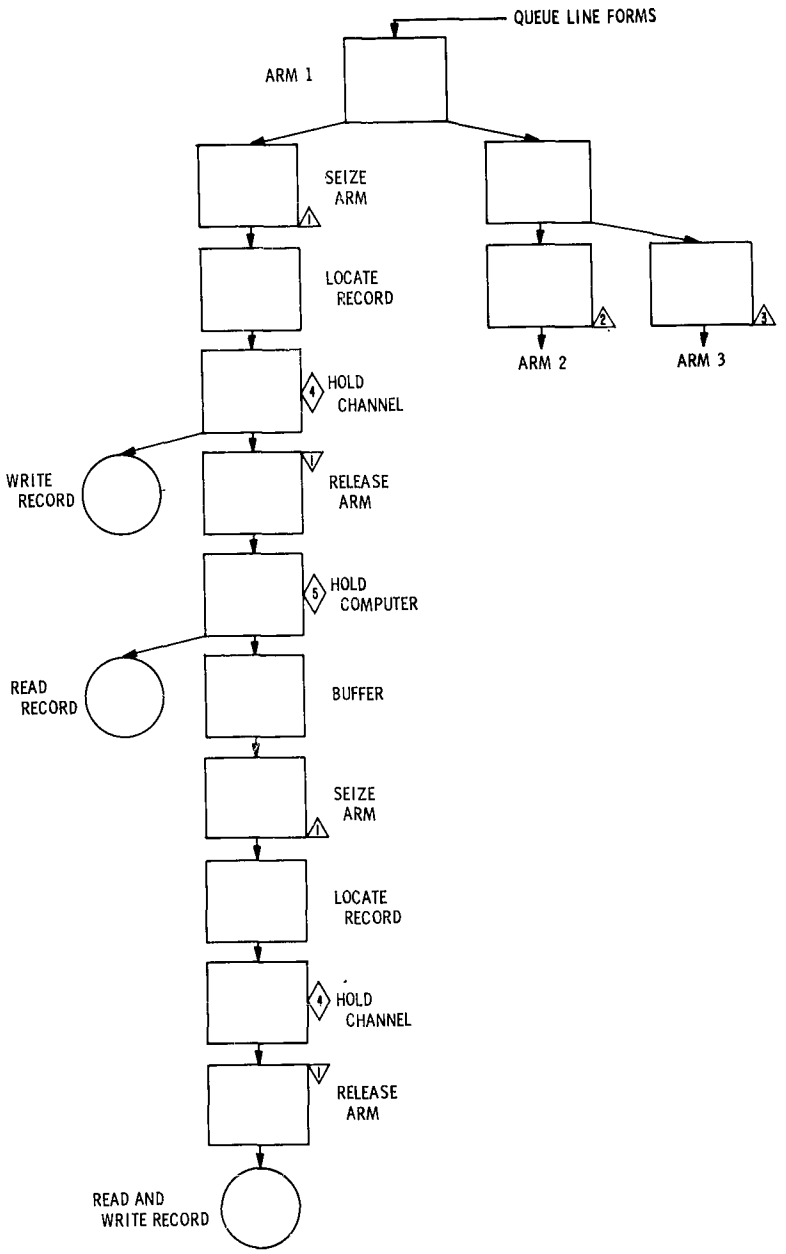


Figure 3. Access to Disc File

The first task to be performed is to seize an arm. Here I am distinguishing between the point at which you take up the arm and the point at which you release the arm. Once you seize the arm, you locate the record. The time in the "locate" block represents the length of time it takes to get the arm and position it. This will then represent the length of time taken to find the record on the disc. The indefinite length of time is dependent upon where the record happened to be, i. e., what part of the cycle it happened to be in.

When the record is located, you need to get hold of a channel to transmit the message back to the computer. Once the message has been returned to the computer, the arm is released. If there is any other transaction waiting for the same arm, then, at the moment the arm is released, the next transaction will immediately proceed and seize the arm. Having transmitted through the channel, you need to hold the computer in order to do some computing. At this point, the write-only record will be removed and you will buffer if you want to come back to do some processing. Some of the messages representing the read-only messages will be taken out at that point and will continue on, repeating this procedure of seizing, locating, and holding the channel.

It can be seen that many other system designs could easily be arranged. The effect of a multi-line channel between the computer and the disc file can be arranged by redefining the capacity of that item of equipment. The effect of having separate channels for each arm or separate channels for input and output can be arranged by defining separate items of equipment for each new channel. By rearranging the blocks, you can put together a system in which, for example, the arm is held for the entire cycle of reading, processing, and writing, instead of being released and re-engaged. If desired, you can arrange to look at a system that requires the channel to be used to address the arm before locating the record. At other points, it would also be possible to expand considerably on the representation of the processing carried out by the computer.

I think that with a little imagination you can represent different systems by simply juggling the blocks. This type of capability is much more in line with what

occurs in system design. In many instances in system design the original design specifications are changed in what appears to be small items of design (for example, going from a half duplex line to a duplex line). If you had written the simulation program from the very beginning, it is quite likely that that change means rewriting the entire program. In this program it is a fairly simple matter to make a change in structure of that type.

As one last example, I would like to cover a point that somebody mentioned very briefly. In this case we will consider the process of reading and writing in more detail. Suppose the system is one in which the channel must be available at the instant the disc is correctly positioned for reading or writing. If the channel is not available, the transactions must wait a time corresponding to a full disc revolution before another attempt can be made. Figure 4 illustrates a block diagram that would replace the single blocks of Type 2 that are being used to represent the action of locating a record in Figure 3.

In Figure 4 a transaction passes through a block of Type 4, representing the seizure of an arm. It takes a time, $M_1 \pm S_1$, to position the arm where M_1 and S_1 are the mean time and spread associated with locating an arm. The transaction passes to a block of Type 2 where it spends a time waiting for the disc to come into position. If T_R is the revolution time of the disc, the mean and spread at this block are both chosen to be $1/2\ T_R$ so that the time for the transactions to advance varies uniformly between 0 (zero) and T_R, thus corresponding to random positioning on the disc. The instant you decide to position the arms, the record could be anywhere; you may have just missed it or it may be just coming up. When the transaction leaves this block, it passes into another Type 2 with zero time. This block acts as a buffer and since it has zero time, immediately transmits the transaction forward in an attempt to engage the channel represented by block Type 3. However, if the channel is busy, the transaction is sent to another Type 2 block to wait a time T_R (one disc revolution) before trying again. The transaction will move back and forth until finally the correct conditions are met and it manages to get the channel at the same

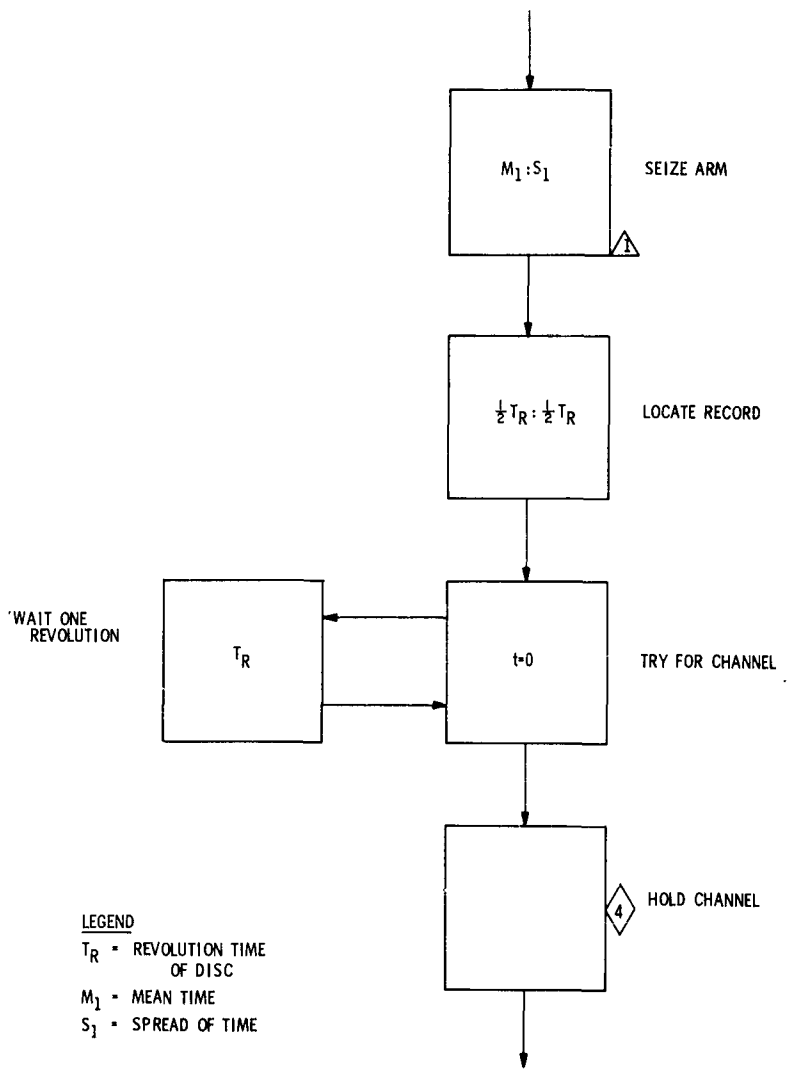


Figure 4. Reading or Writing Records

time the record is there. This, I think, is a good example of some of the decision procedures the program will accept.

These examples have been based on a simple program with only a few block types. They illustrate, however, several of the main advantages to be gained from general-purpose programs. In a short span of time you can write down the description of a system in a block diagram form, almost as you are attempting to describe the system. Without any reference to the program, you can have an exact description of the system and the program will have to accept this description. The program does so by having separate sub-routines representing the block diagrams. All you need is a mechanism for assembling the various sub-routines and going through the process of moving transactions through the program.

A second major advantage is that this method of building models is flexible enough to allow the model to expand or contract with extreme simplicity as the system being studied changes. In the example, we saw that we were able to take certain sections of the system and blow them up for further investigation.

One of the unexpected advantages of this approach has been that it imposed a discipline upon the users in dealing with a basic language. The discipline of having to think out carefully and to express what the precise actions are as represented by a particular part of the system, is beneficial in system design work. As you approach this ideal of a system language, you even can begin to see similarities between apparently different systems. You begin to see that the logical structure is the same.

Although we can describe a wide variety of different types of systems with such a program, it is necessary to increase the number of basic block types that have been employed in these examples, in order to get efficient models with better details. If you try to describe your sample in terms of these basic block types, eventually you arrive at some feature of the system which cannot be described in this way. As you think about why you cannot describe it in this fashion, you will find yourself automatically defining another block type that will enable you to do whatever the action is. In this manner we have extended the block types in our program to the point where about twenty block types seem to be adequate for almost any type of system.

General-purpose programs of this type have proved to be efficient in terms of machine time, but, perhaps most important of all, general-purpose programs built on the principles described in this paper have considerably increased the speed with which simulation results can be obtained. They can turn simulation into an everyday tool for system study.

Mr. Gordon has presented similar material in a formal paper which has been published in Computers: Key to Total Systems Control, Vol. 20, Proceedings of the Eastern Joint Computer Conference, 1961, Macmillan Co.

ADDITIONAL COMMENTS: GEOFFREY GORDON

ADAMS: Geoff, would you go through the "push-cart" example to demonstrate the use of your program to handle a particular problem?

GORDON: The following is an example of a supermarket problem:

When shoppers first come into the store, they must get a basket. Since there are a limited number of baskets, the rule will be that if they don't get a basket immediately they will wait, but only for a certain length of time. If they are forced to wait too long, they will become exasperated and leave the supermarket.

When they get a basket, they shop, and then queue up at the counters. Two types of shoppers will be distinguished: the general shopper and the express shopper. The distinction is that the express shopper can go to one special counter and the remaining shoppers can go to one of three other counters.

Figure 5 shows how a system of this type can be represented. Again there is an originate node which is used to create transactions; in this case a transaction is a human being. The transaction passes into a buffer to try to get a basket. Baskets, as a set, will be defined as an item of equipment with a given capacity. Unless the shopper can get a basket immediately, he goes into another buffer to wait a given length of time and then tries to get a basket. If the path is still closed, the shopper waits once more and then tries again. If he can't get a basket this time, he will go to a terminate node and leave the store.

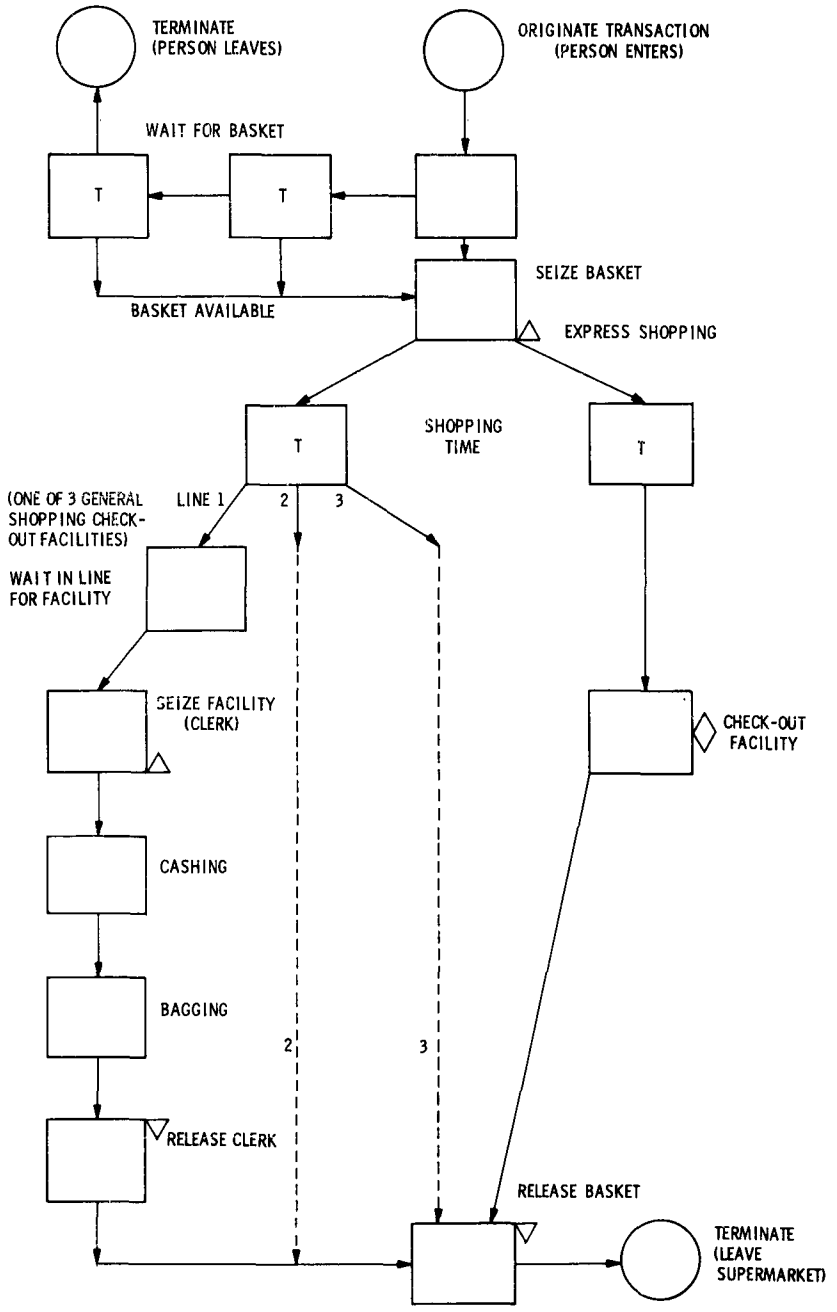


Figure 5. "Push Cart" or Grocery Store Problem

GORDON: (cont.)

If a basket becomes available before the end of the waiting time, the shopper "seizes" a basket and starts shopping. In the diagram, the traffic is split into two paths; one for the express shoppers and one for the general shoppers. The shoppers first go through an "advance" box which merely represents the shopping time. Here there are two different times; one for the express shopping and one for the general shopping.

In this system, the clerks who perform the check-out service are represented as facilities and thus are items of equipment with the transactions (shoppers) engaging their services for checking out. As there is only one express clerk, this facility is represented by one item of equipment. The facilities for the general shoppers can either be lumped together as one item of equipment with a capacity of three, or can be separated into three streams and we can study the behavior of each.

For this case I have created three separate streams with a separate clerk for each stream. I have made one distinct assumption for the general-purpose shoppers; the actions of cashing and bagging are separate tasks.

First of all, then, I "seize" the service of the clerk, spend some time cashing, some time bagging, and then "release" his services. Once bagging is finished, I can then release the basket and leave the store. Obviously, I could lump the separate boxes representing cashing and bagging into one box, but it's more convenient to show them separated. The express line is represented differently as I have assumed that there is no need to do any bagging, since there are so few items in the shopping. All you need is to "hold" the service of the clerk long enough to do the cashing. Once the cashing is done, the basket will be released and the shopper will leave the store.

GORDON: (cont.) Thus, the two examples I have given illustrate rather well how these concepts can be applied in a large number of ways. In the first system (Figure 2), the transaction was a message; in this second system, the transactions are human beings. You can also see how general the concept of equipment is. With a little ingenuity, this basic language can be applied to a wide variety of systems.

I do not want to give you the impression that this language can only be applied to small-scale problems. I just used for examples problems that had only ten to twenty boxes. Actually, we have built simulations which have had as many as two thousand boxes. We have not built larger programs as I feel that when you reach that size in a simulation program, then you must stop and do some proving in your system. Too many people tend to represent all the little details in the system and finish with a structure that has great detail in some parts and only gross tones in other parts. Our program is one that gives you the ability to expand and contract very easily and I believe that it should be used in that way. By the time you reach two thousand blocks, it is time to consider whether or not you have too much detail.

BURROWS: I would like to ask two related questions. First of all, it appeared that as you were taking the four types of shoppers (transactions) through the system shown in Figure 5, you could give the percentage of the load represented by each type of shopper. At some point in the program it may turn out that the path of servicing followed by the separate types may split, later come back together, and then split again later. If you randomly make the decision at the first split, as to which type of shopper a particular shopper is, you won't have the same load between the two areas in the time period after the second split if you again randomly make the decision. Do you make any provision for remembering the first decision in order to keep a consistent treatment of the load through the whole system?

GORDON: This is handled by block types which tag particular transactions. It's quite easy to arrange the specific job of one of the block types to tag the transactions so that in a later block type you can identify which path was previously followed. This is quite similar to a "TSX" instruction. The transaction can carry a note which says "I went by way of route one." I only mentioned two mechanisms for deciding which way you should go; one was random choice and the other was alternative selection. The third mechanism for determining the route to follow is presetting a previous node which could have notated the transaction.

BURROWS: You've answered my second question which was: "What description of a transaction is allowed at the beginning and can you erase these descriptions?" These descriptions essentially determine which side of the gate the transaction goes through. Can you erase these descriptions and add other tags as a transaction goes through the system?

GORDON: You can put a tag on the transaction which says: "The next time you are asked, 'Which way shall I go?', look at this note and it will tell you." Once that action is performed the tag is set to zero and is ready to be reset to another value. When a transaction is tagged at a particular box, the tag can be considered as the mechanism for choosing a successor. The tag is erased as the action of the tag is performed. If you want to preserve the tag, you must reassign it; if you don't want to preserve the tag, it's ready to be used for the next decision.

BURROWS: So the description of the tag is essentially comprised of one bit. Since you only have two-way nodes, does the tag simply decide whether to go one way or another, or is it deeper than that?

GORDON: Actually the tag is an absolute address. The box which makes this decision uses the tag established for the transaction as the actual address to which the transaction is to go.

GORDON: (cont.) In other words, if one must tag a transaction when the decision is made, it is simplest to tag the transaction with the number of the node or block to which it will eventually go. This could be done, as you have suggested, by arranging a single bit to say, "Go this way" or "Go that way."

BURROWS: In an intricate processing system the number of descriptors or adjectives that describe each transaction is quite large. Future processing depends upon this description and, indeed, some decisions made during this processing will change later on, if it is delayed too long in one place.

GORDON: Although I've only mentioned one descriptor, other descriptors can be attached to a transaction. Any of the properties that I've been discussing can be made functions of these descriptors. For example, you can attach a descriptor to a transaction such that the time of that particular box is a function of the descriptor.

When you include descriptors though, the program gets more complicated. Our type of program is quite straightforward if you keep within this partial flow type of simulation where the transactions do not have individual identity.

BURROWS: Is it fair to say that the aspects of the system you study are the effects of the cascaded waiting lines, the number of servers and the priority schemes?

GORDON: Yes. I would say that the best single item we get in the systems we have studied is the delay time. What we want to measure is the percentage of messages that are handled in a given time so that we can derive a particular distribution. With this we can determine the probability of being delayed more than a certain amount of time.

BURROWS: It's not qualitative in terms of the adequacy of the service except in terms of delay time.

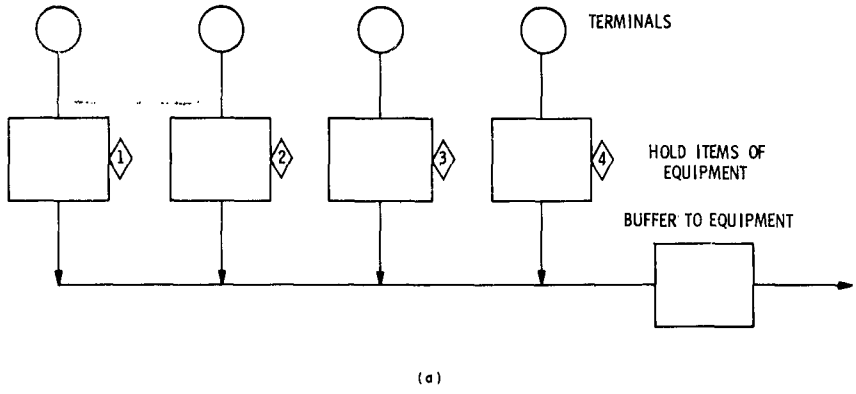
GORDON: Well, there is some relationship between adequacy of service and delay time. If you can decrease the delay time then you can improve the service.

Of course a second factor is the information regarding the utilization of the equipment. It becomes quite obvious whether or not you have provided enough lines or enough computing capacity or enough storage.

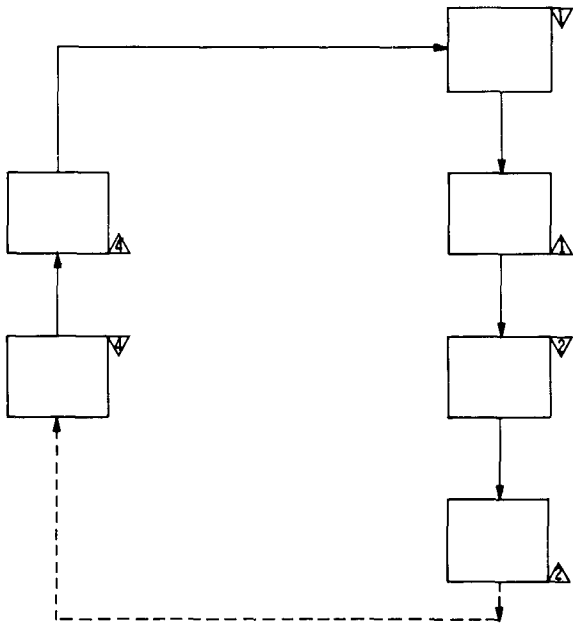
MOLNAR: In many instances, critical decisions cannot be represented by choice points because the conditions keep changing and a man must determine whether or not a decision has to be made. Besides, he must diagnose what alternatives are available prior to the choice point.

GORDON: It is possible to describe two systems. One is the operating system and one is the decision system, both of which interlock. As a simple example of this idea we can look at a common problem in communication work. A number of terminals share one line and only one terminal at a time can use the line; a mechanism can be established for transferring control from one terminal to another.

Figure 6(a) describes a technique that can be used for representing a system like this. We have a series of originate blocks, each of which represents a terminal where messages are created. We send each of the separate messages into a hold block and if a message gets through the hold block, it is then equivalent to saying, "The message is on the line." Once on the line it goes wherever you desire it to go (e.g., a computer).



(a)



(b)

Figure 6. Simple Communications Network

GORDON: (cont.) Now we build another network shown in Figure 6(b) in which we indicate the items of equipment. Assume that they all have a capacity of one; i. e., only one transaction can be handled at a time, and there is a closed loop in which you release equipment "No. 1," and then reseize it after a pause. You then release equipment "No. 2" and reseize it after a pause, and so on. The loop can be initialized so that when the transaction has been injected into the loop, it has already seized "No. 1," "No. 2," "No. 3," "No. 4," etc. Thus, this one single transaction has control of all the equipment. Because the transaction has control of all the equipment, it blocks transactions in Figure 6(a). When the control transaction begins a loop, it releases "No. 1" which opens the line in Figure 6(b) to transactions to line 1. After a pause, the control transaction shuts the line off. Then it opens up "No. 2" and shuts it off; opens up "No. 3" and shuts it off, etc. So this single transaction is following some type of decision process which can get as complicated as you wish. The conditions under which you open or shut these paths in Figure 6(a) can vary; e. g., it might simply be a clock pulse sending the control transaction around, or it might be a very elaborate decision procedure by which it is decided whether or not "No. 1," or "No. 2," or "No. 3," etc. is opened. It is feasible, then, to build up fairly elaborate decision procedures inside the program by techniques of this type.

The concept of an item of equipment is used here to provide control that interlocks two different systems; viz., the operating system and the decision system.

ADAMS: We may be using the term "decision" when we actually mean "selection." The methods that we have been discussing so far are valid for those classes of decisions that are only selections of alternatives.

GORDON: With these programs, though, you can get a high level of decision making. In Figure 6(b) the decision can be determined by the traffic currently going through the system or that has gone through the system. If the traffic is exceptionally high on "No. 3" and is exceptionally light on "No. 1" (given a definition of "high" and "light"), then within specified bounds "No. 1" is shut down and the time originally allotted to "No. 1" in the control loop is given to "No. 3."

In Figure 6(a) the systems could be a continental link system with Station No. 1 in San Francisco working eight hours a day and Station No. 4 in New York working a different eight-hour period. One condition could be that, if the time of day is in a certain period, the control transaction skips Station No. 1 and goes to Station No. 3. In other words, No. 1 is left permanently shut off for a particular time period.

How do you know what time of day it is? Well, you have another system representing a transaction ticking around. If, in the closed loop shown in Figure 7, each one of the boxes represents a time duration of one hour, then, after one hour, the timing transaction will move into a new block. Timing control can be exercised by the technique of equipment. When the timing transaction has gone through a block after one hour, an item of equipment could be seized by the timing transaction. This can place a block in front of the control transaction and force it to choose an alternative path. In other words, the conditions under which the control transaction proceeds are that: If the timing transaction is in one part of the loop, then the time of day is a certain figure, and one route is chosen; otherwise, an alternative route is taken. In this manner you can have many levels of decision making. However, it does become awkward, since you can no longer see exactly what each of these networks represents.

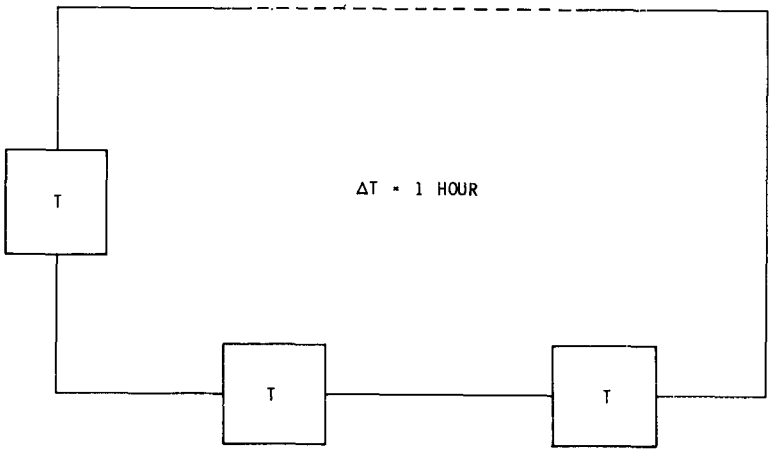


Figure 7. Time Control Loop

II

A FUNCTIONAL MODEL AND ITS UTILIZATION IN THE DESIGN OF A COMPLEX SYSTEM

Irving A. Wallach

System Development Corporation, Paramus, New Jersey

Due to a change in Department of Defense requirements, we were unable to obtain a clearance for public dissemination of this paper. It will be released and published as a Supplement to this Report at a later date.

III

GENERALIZED MODELING OF COMPLEX SYSTEMS*

Harold W. Adams
The MITRE Corporation, Bedford, Massachusetts

The concern of this paper is the attempt to develop a generalized model of complex decision systems; in particular, the Air Force command and control systems. Some of the general concepts and the approach we have taken to this problem are discussed, as well as some details of a tentative system model.

The job of the Air Force in a command-control system appears to be the management of the force. The decision problem is thus a resource management problem: Where are you going to put your resources so that they will do you the most good in terms of what you want?

It seems to me that one of the main differences between the military and the industrial situation in many of these management systems is that, in general, the industrial system is set up to operate on some sort of presumed stable, steady state, whereas, in general, the military systems are set up to operate under crisis conditions of one sort or another. Exactly what this differentiation means in practical terms, I don't know. I do feel the point is one worth some study and clarification, in view of the success some have reported in developing generalized models of a business.

Another point that we should concern ourselves with is the matter of what makes a system complex. The closest I can come to it, and I am not at all happy with this as a definition, is that a complex system is one that does more than one thing. By more than one thing I mean that it may at one time, in one state, do a certain

*This study has been previously reported in a document by H. W. Adams, C. M. Festa and J. G. Robertson entitled, "The Functions of a Decision System," MITRE Working Paper W-3856, April 7, 1960.

thing and may at another time, in another state, do a different thing.* There can, of course, be all sorts of combinations of such system functions.

First, let me indicate the large degree of similarity between the approach that Geoffrey Gordon has taken and the approach that we have taken. The similarity is not coincidence. We are quite frankly trying to apply his technique to our problems with complex systems. To give an idea of our goal, let me quote from a paper of August 1960 written by Gordon and three others.

> Often problems arise in late stages of system development which could have been avoided by better decisions at the beginning. At the outset, the designer cannot recognize the implications of his choices because of his limited knowledge at the time the decisions have to be made. If the vicious circle is ever to be broken, a method for studying the performance of rough "paper designs" is needed.**

This is approximately what we are trying to do here at MITRE. The authors of the paper indicate that "micro-simulation," the simulation of the detail of the system, is impossible early in the system design process because we know only the broad outlines of what the system is to be. One could not, at this point, specify detail if he wanted to, except in rather exceptional cases where he just happens to know certain details. Since the system is known only broadly, it may therefore be less complex to simulate at this stage. That is, we may be able to simplify our model without loss of real validity.

The paper then describes the work they have done in generalized simulation modeling. In his paper, Geoff (Gordon) has told you of this approach. We

---

*See also the discussion in Newell and Simon, "The Logic Theory Machine: A Complex Information Processing System," The RAND Corporation, D-868, July 12, 1956 (p. 2).

**D. L. Dietmeyer, G. Gordon, J. P. Runyon, and B. A. Tague, "An Interpretive Simulation Program for Estimating Occupancy and Delay in Traffic-Handling Systems which are Incompletely Detailed," AIEE Conference Paper 60-1090, Pacific General Meeting, August 8-12, 1960 (p. 1).

are trying to see how far we can get using an approach like Geoff's and applying it to a problem like the one that was outlined by Irv (Wallach).

Hopefully, the model we are developing can be used for rapid and reasonably precise representations of system design alternatives which could then be subjected to simulation analysis. The analyses thereby derived could be applied to the recurring MITRE problem of evaluating alternative system design proposals, both those of contractors and of in-house personnel.

Our approach to the system is characterized by concentration on system functions. Here again, we have followed the lead of Geoff Gordon by concentrating on the action verbs which specify the system functions. A major virtue of this approach is that it avoids concern with system equipment at the stage when such concern is both inappropriate and often harmful.

The most abstract representation of our decision system model is shown in Figure 1. A closed loop exists in which your system and the real world have an ongoing process of interaction. The process starts with a real world which you desire to have some particular structure; you want it to have a structure favorable to you. Your sensors tell you how things are in the real world. After you clean up the sensor input data through data quality control, you make some sort of assessment of the situation: Is the world as you would like it to be? If it is not, some decision must be made about the action or lack of action you wish to take, so as to restructure the real world more along the lines you would like to have it. Thus, the process cycles; a decision is made, an action is taken, presumably the real world in some way is changed, the sensors and further assessment then tell you whether or not the change is as you want it. We have a continuously cycling closed loop.

The next problem is to develop a more concrete model. From a theoretical point of view we are convinced that the problem must be approached deductively, with inductive checks as we proceed. The next step is, therefore, to model the internal structure (the logical and sequential flow of functions) of the two fundamental "system action verbs," ASSESS and DECIDE. (The data quality control function in which

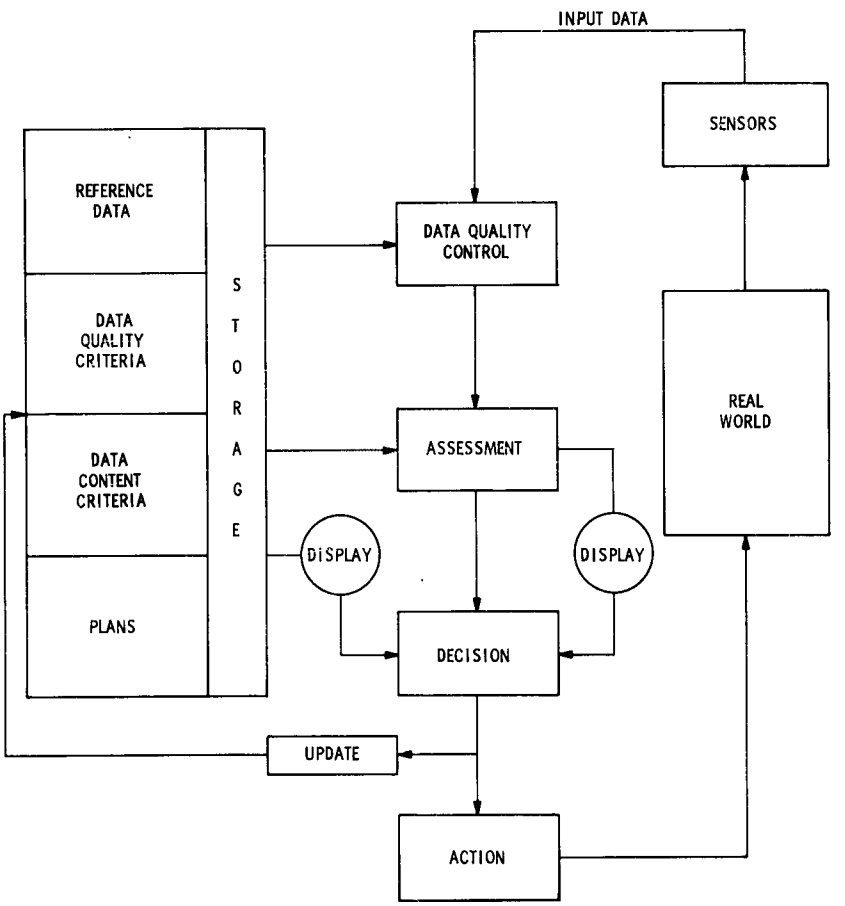


Figure 1. Generalized Decision System Model

the input data is cleaned up has been excluded from consideration not for reasons of unimportance, but rather because it is not a prime functional operation for our purposes.) The reasonable way to do more detailed modeling seems to be to describe each of the major functional verbs by the lesser verbs which indicate the subprocesses which make up the whole. If we proceed along this path to successively higher levels of specificity, we should soon arrive at the point where a computer program can be written which will simulate the function represented when suitable constants and parameters are inserted.

One approach to the ASSESS subprocess is that shown in Figure 2(a). The ASSESS function asks the question; "What is going on?" and is seen to be composed of the three subfunctions: monitor-monitoring what is going on based on sensor inputs; compare-comparing what you found about the world with the picture you have in storage of what you desire the world to be; evaluate-evaluating whether there are differences between the real and desired worlds and, if there are differences, whether these are significant to you, and, if they are significant to you, whether you wish to take action. If so, you must then DECIDE what you want to do to minimize the difference between the picture of the world as it is and the picture of the world as you would like to have it. *

The abstraction level is still too high. Figure 2(b) takes one part of the ASSESS function, monitor, down another abstraction level. The subfunctions of monitor are seen to be, in order; scan, select, categorize, correlate. At this level we find we can already indicate the points at which information flow elements have to be brought in, we can specify some of the decision loops, and we find that binary choice points begin to appear.

*The process outlined here is conceptually similar to that of Newell, Shaw and Simon's work on general problem-solving programs. A. Newell, J. C. Shaw, H. A. Simon. "Report on a General Problem-Solving Program," The RAND Corporation. P-1584. Dec. 30. 1958. Rev.. Feb. 9. 1959.

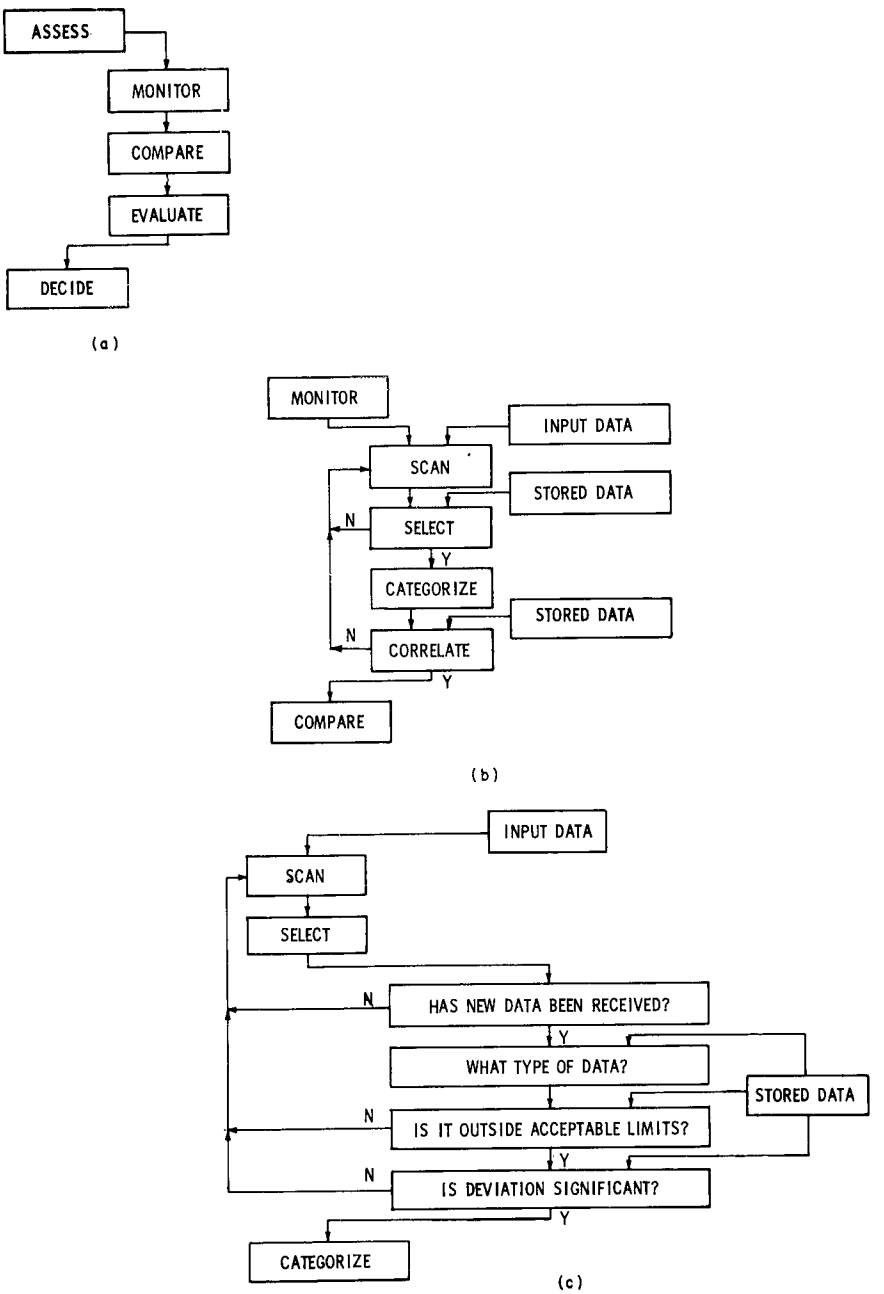


Figure 2. Functional Flow Diagram of ASSESS Function

We have tried to indicate that select can be analyzed into the four questions indicated in Figure 2(c). Again we have decision loops and binary choice points. We also have reached the point where the binary questions we are asking are nearly real-world-specific in their content.

On completion of the ASSESS function, we have found that either the real world is as we want it to be, in which case no response is required on our part; or the real world is not as we want it to be, in which case a response from us is required to redress matters and achieve our goal. The DECIDE function is concerned with selecting from the universe of possible responses that one which seems most likely to achieve our goal.

When we enter the DECIDE function, we know where we are, we know we don't like where we are, and we know where we would like to be. Our problem is to determine how to get to where we want to be. We have taken a rather straightforward game theory approach to this problem. We assume first that we have a multipurpose system. We then model the DECIDE function in the manner indicated by the functional flow diagram shown in Figure 3.*

Initially we must compare the problem and the goal with the response classes stored in our machine. We assume that there are 1 through n response classes stored in the machine from which a selection could be made.

By select applicable response classes we conceive response classes as a selection of some "punishment suited to the crime." For example, if the situation is that the Soviets are sending troops to the Congo, somehow you feel this is not necessarily the occasion for firing the Minuteman missile. The more applicable response class would seem to be, "Send American troops to the Congo."

*The sequence diagram structure draws heavily on an analysis developed by Dr. Joseph G. Robertson, Jr., of MITRE.

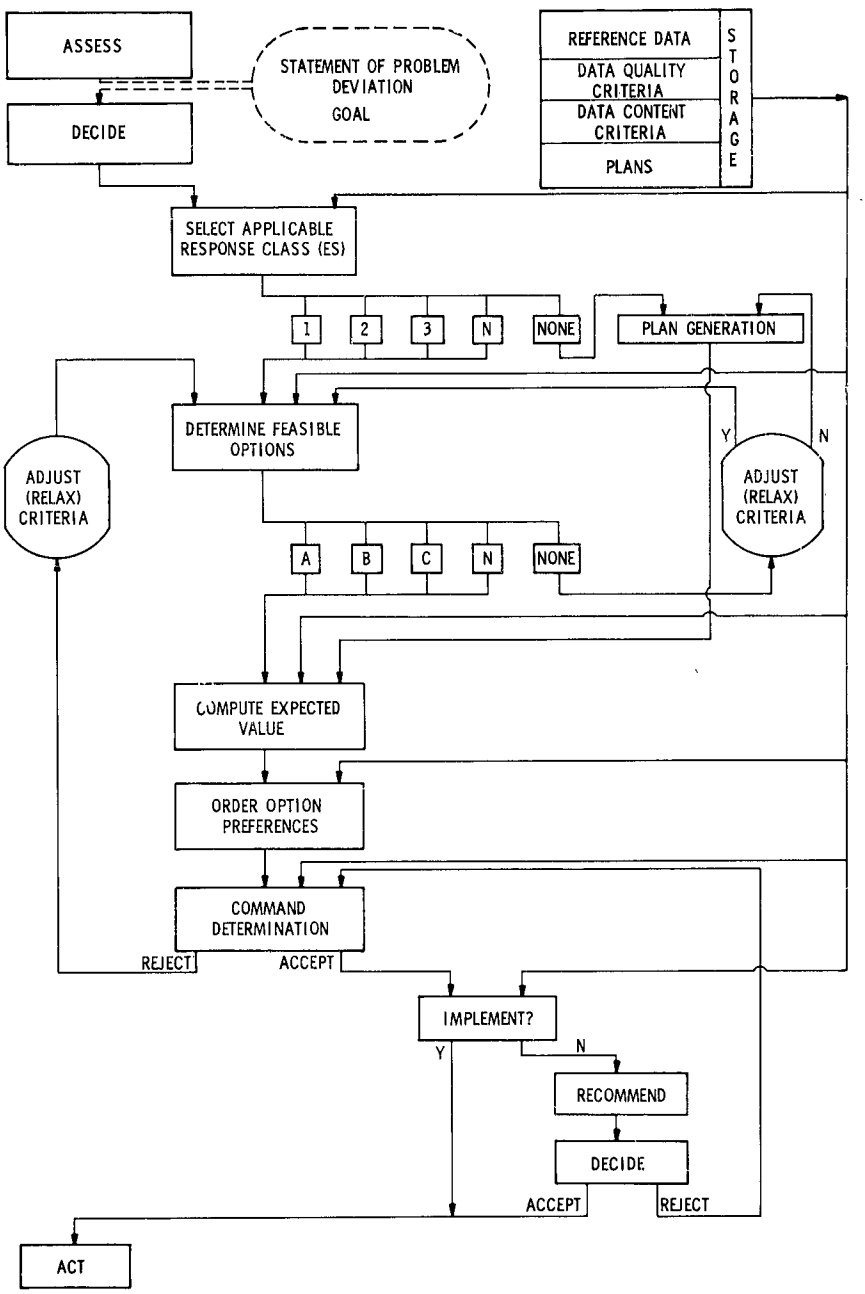


Figure 3. Functional Flow Diagram of DECIDE Function

If none of the response classes seem appropriate to the situation, then you go into plan generation. This box, as far as we've been able to take it, is unstructured; we simply do not know the internals of this function. We would probably suggest that plan generation is a function best left to people in the command staff somewhere off-line of the machine. This manual plan generation subfunction will, as we view it, produce both a response class and the feasible options within this response class.

Once we have found response classes which are applicable, we must then determine the options within these classes which are feasible for us to pursue in the light of our resources. If we find that one or more of the options within the classes are feasible, we must then compute the expected value of each option. If there are no feasible options, a determination must be made of whether we are able to relax the resource criteria (have we been authorized by our commander to do so?), or whether, criteria relaxation being impossible, we must generate a new plan.

If it is within the authority of the system at this point to relax the criteria, you go along the "Yes" route and take another look at feasible options to see if something is now feasible in view of your new criteria. If you cannot adjust your criteria and you have found that none of the feasible options will satisfy the goal, you then go back to plan generation.

Given a set of feasible options, the expected value of each of the options in terms of your goal is computed by the rather standard game-theory approach indicated in Figure 4. This is basically a cost/value tradeoff. The result here will be a series of option value statements (e.g., "Will this option achieve our goal and then some?" perhaps indicating it requires an excess of resources for the task; "Will that option barely achieve the goal?" perhaps indicating the risk of not achieving the goal should implementation be imperfect in any respect).

Given these option value statements and comparing them with stored criteria, we can order the options according to preference criteria indicated by the commander. We are then able to present to the commander for his

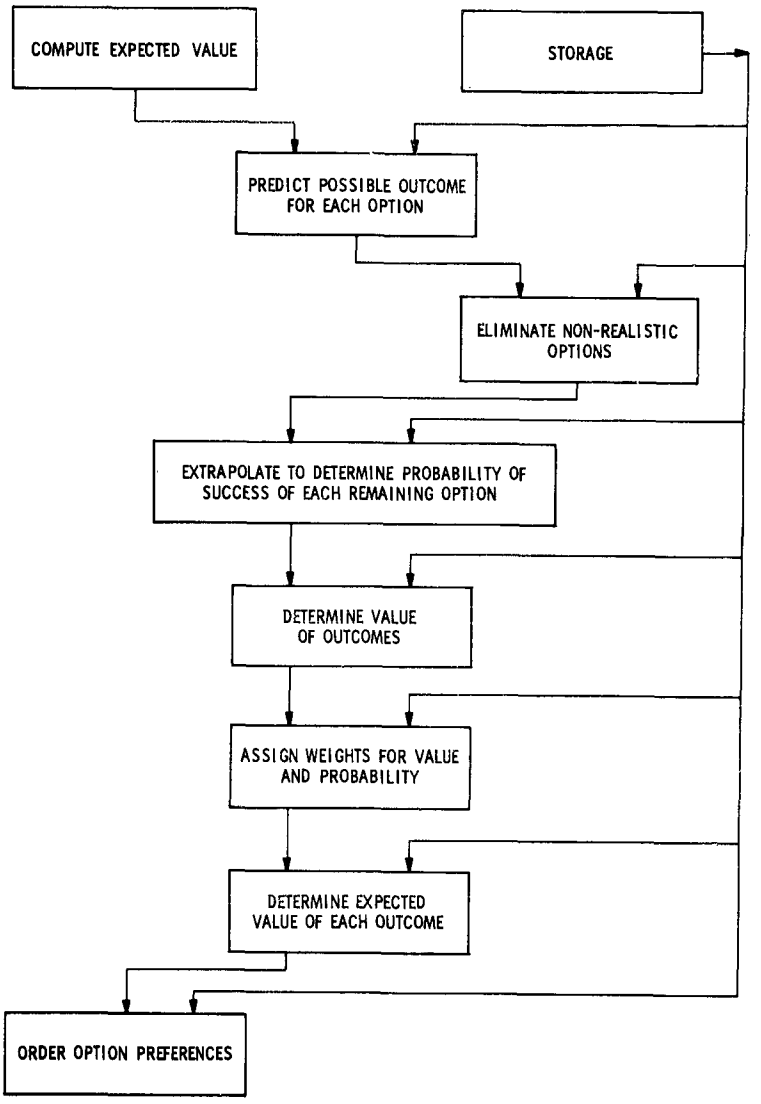


Figure 4. Specific Breakout of Compute Function within DECIDE

determination, the preferred options which appear appropriate with a preference ranking of these options.

The commander must then determine either to reject all options, or to select an option he feels should be implemented. If he rejects all, he will indicate how option-assessing criteria should be adjusted so that the decision process preceding his determination will function more in line with his view of the situation. This done, criteria are adjusted and feasible options are selected on the basis of the commander's restructuring.

If the commander selects an option he feels should be implemented, the question becomes: Can he implement this option within the constraints imposed on him by higher command? If he can, he does, by issuing action directives. If he cannot, he recommends his preferred option to his superior. If the superior commander accepts the recommendation, action directives are issued. If he rejects the recommendation, he indicates the basis for the rejection; that is, more clearly specifies his preference criteria. The problem then goes back down an echelon for subordinate command determination of whether to select another option from those previously presented in rank order or whether to reject all these options and recycle.

Here again, the only conceivable reason that you would recycle is if you are able to adjust some of your criteria in the sense of relaxing them or of changing them in a way which would open up alternatives not previously feasible.

Now, at a certain level of concreteness, a system model or a functional building block becomes meaningful in terms of the constants, parameters, and other factors which affect the operation of the system. If you can take these functions down to some at present undetermined level, you somehow feel that you have reached the real world.

The problem once more is: At what level of concreteness do we know the facts about the system? This gets back to the original point I tried to make with the quote from Geoff's (Gordon) paper. In the early design stages, when a model such as this might be most useful, you really don't know much concrete about the system. This,

in a way, simplifies your modeling problem because you cannot go very far into the detail. It seems to me that if you can't go into much detail, then some of the arguments against abstract modeling are undercut—undercut because you have no choice except either to try your system design in some sort of a model or to leave it to the best judgment, intuition, experience, whatever you want to call it, of the people suggesting the design.

Now it may well be that this judgment is superior to a simulation model. In fact, if you had the right people there, I dare say it would be. However, I suggest that from a very practical point of view, a model like this can, given the frailties of people, be very useful in two senses; first, it can help build up the understanding of the people who are concerned with the effect of the design decision, and therefore, the confidence they have in the conclusions they may have reached in other ways. The second point is one that William Haythorn of The RAND Corporation Logistics Laboratory has often made. Bill, as you may know, has run a series of simulations on base maintenance of missile squadrons. He has found that one of the beauties of simulation is that, if you simulate something and come to a conclusion, the fact and the paraphrenalia of simulation tend to heighten in your buyer's mind, in your boss's mind, or in the Air Force's mind, their credence in the conclusions you have reached. It becomes easier for them to believe that your decision is the right decision if you say that you've put this through a thorough simulation. It is scientific in approach and has cost a lot of money; therefore, "it must be correct." In any event, I think that in the give-and-take of the political world that exists, this type of credence level for a customer may be a valuable side benefit.

It is also worthwhile to point out a potential defect in computer simulation analysis and decision. Some of you may have been following the MIT Centennial Lecture Series on "Management and the Computer of the Future." Norbert Weiner, speaking of his fears about computers and their future use, made this point:* He

*Symposium on Scientists on Decision Making, May 5, 1961, MIT, Cambridge, Mass.

fears that, excepting a few rather strong personalities, a tendency may develop for the computer to take decisions out of the hands of the people responsible for making decisions. Those charged with making final decisions may someday think it's simply enough to say that the computer says thus and so, therefore this should be done, and further, that the computer can be responsible for any mistakes which may result. This would essentially erode the base of any responsibility the people may have felt.

Weiner underlined this point by making the analogy that to use computers to make decisions is a sort of "Russian roulette for managers." As he sees it, Russian roulette is, psychologically, simply a game which may enable you to do something you want to do without taking any responsibility for what's been done. If you happen, in the process of the game, to kill somebody, that's not your fault, it was just the result of a random process which discharged the bullet.

Now, to return to the beneficial side of simulation models in system design, let me repeat a point made by Irv (Wallach): the model imposes an overall logic upon the system. If the model hangs together, and if, when you turn the crank of the machine, a simulation runs through properly, you feel that you have an integrated and self-consistent system. The really important step in system design is thus to specify the system functions into manageable packages within the overall structure of coherence given you by your model. To put it into simpler terms, if we have isolated the functions of a system, it is necessary, then, to define the tasks that must go on within these functions so that the functions will be fulfilled. It is this level to which one should attempt to go in the development of a general model.

That is roughly where we got, on our first approach to the general system model. At this point I feel it appropriate to read the caveat we inserted at the end of our discussion of our model.

> Finally, let us inject a note of realism. Others who have pursued the general system simulation model have had to accept a compromise to the original ideal of a single model. Very simply, a single model either had to have an excessive number of parts which would lie unused in any particular simulation application, or else the single model had

> insufficient generality to cover the various facets of the real world encountered in the many systems to which the model was to be applied. The obvious answer, therefore, is to develop a library of basic system building blocks—each with its own program, each covering a function found in many (though perhaps not all) systems—from which specific system simulation models can be put together rapidly as required. *

With this first attempt at a general simulation model for command and control decision systems, it became fairly obvious that very little more was to be gained by sitting back in our ivory towers and analyzing a pure and logical world. We felt we needed to try out our model in a particular system and see how much overlap there was between our logical diagrams and the real world. The problem chosen was the transport problem in the USAF Command Post. We chose the transport problem because we could think of no problem which was better worked out as far as the techniques were concerned, and about which more was known, or for which more samples of previous solutions were available. A more immediate reason for choosing this problem was that Peter Ten Eyck and Charles Morrill, both in our department, had already made a model which simulated the transport problem. The diagrams shown in Appendix A indicate the degree of overlap we have found between the transport model and the general model I have discussed here. Let me emphasize that none of this has been put on a machine. It all occurs in the real world only in the form of these diagrams and we have not reduced the diagrams to a computer program.

In the following paper, C. S. Morrill and P. H. Ten Eyck discuss the specific case we used and the degree of overlap found between the transport problem and our general system model.

---

*H. W. Adams, C. M. Festa, J. G. Robertson, "The Functions of a Decision System," MITRE W-3856, April 7, 1961 (p. 4).

IV

CASE STUDY: SIMULATION OF A LOGISTICS PROBLEM

Charles S. Morrill and Peter H. Ten Eyck
The MITRE Corporation, Bedford, Massachusetts

Before delving into the application of the general model to a specific case, I would first like to present the specific case that was used. In our own study, the basic problem to be simulated was the logistics problem confronting the operations center of an Air Force command and control system. Essentially, the problem is to determine in considerable detail one or more feasible ways of handling a request for MATS transport of goods from one region to another. In the request, a priority level and a time limit for processing the request are specified; and within the problem several sources, cargoes, and types of transport aircraft may be involved. The sequential steps in handling this problem are depicted in the flow diagram in Figure 1 and are described in detail in Appendix B.

This problem resolves, essentially, into three parts, which are represented as sources, destinations, and routing.

Sources refer to the selection of airbases from which aircraft are to be obtained and to the quantities to be drawn from each airbase. It includes the questions of whether the soliciting force is to have its entire request filled, and if not, what proportion of the request is to be filled.

Destinations involve the specification of the bases which are to receive the aircraft, and the quantities which are to be sent to each base. In some cases these may be specified by other headquarters. However, if only part of the request complement is to be moved, this reallocation must be directed by the command center.

Routing embodies, with some embellishments, the classical transportation problem of operations research. The problem is to answer a question like the

following: Given a set of sources, each with some aircraft, a set of destinations, each requiring some aircraft, and a cost (in money, time, or losses, for example) specified for each source-to-destination movement of a single aircraft, what is the least expensive way to send the aircraft?

Some of the assumptions made for this problem are as follows, with the complete list given in Appendix B:

(a) all data received at the operations center are completely reliable;
(b) a source has both aircraft and cargo on hand;
(c) no aircraft will fly without some cargo, and the amount of cargo will remain constant throughout the trip; and
(d) problems of ground transport of cargo or variations due to reloading of cargo from one aircraft to another are not considered.

In this problem, the operations center receives requests from a Requesting Agency (RA), which is always at a higher level of command, and is considered to possess less detailed information than the operations center. Thus, jobs may appear feasible at the agency, but may not, in fact, be feasible.

There are many possible modifications of the sample problem which would do no violence to the logistics model, but would simply alter the solutions. A library of programs would be needed for any extensive simulation program and could be easily generated by varying some of the elements of the problem. Essentially, there are five elements in the request; viz., priority, destination(s), operational requirements and quantity of cargo, aircraft information, and time.

I intend, here, only to indicate the manner in which the problem is handled and how the flow chart in Figure 1 depicts this. The first question asked after the request is received is: "Is the job feasible?" At this step (2) there is an experienced operator who is asked if, according to his knowledge, this request can be fulfilled without going through all the mechanics of the solution. If he is certain that the request cannot be fulfilled, he sends it back to the requesting agency with the

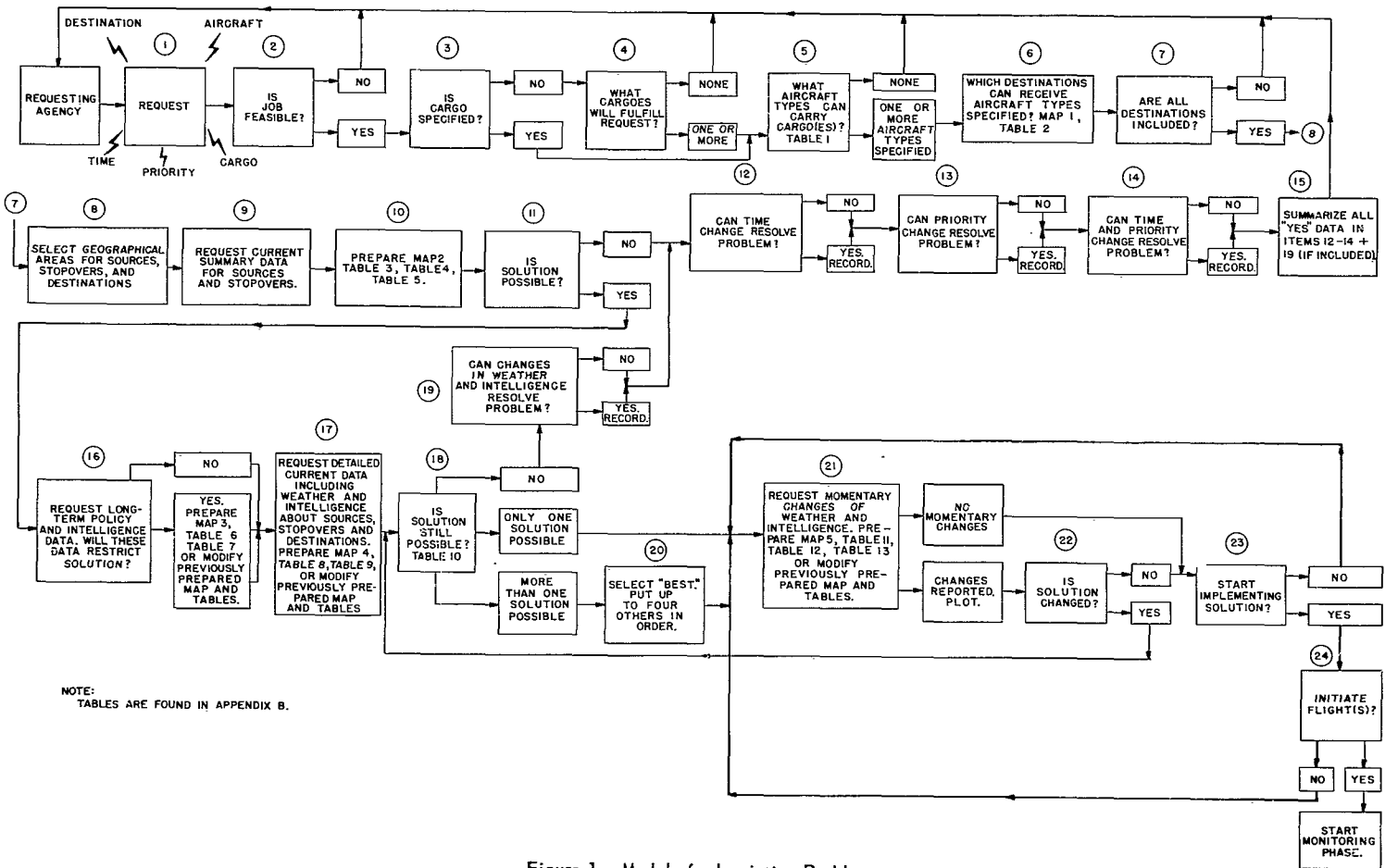


Figure 1. Model of a Logistics Problem

stipulation of why. Everytime it is necessary to return to the RA during the solution of the request, the RA is told what the problem was, so that it can issue a more reasonable request or modify the original constraints of the request. If the experienced operator feels the request can possibly be fulfilled, or a more reasonable request has been issued, the process continues through the remaining steps indicated in the diagram until the problem is solved, or it is necessary to return to the RA. We ask about cargo and what cargoes will fulfill the request, and we ask about what type of aircraft can carry the cargo. At step 7, we ask whether or not all the specified destinations can receive the given type of aircraft. In our particular example, we found that destination 9 could not receive any of the given type of aircraft. Thus, the requesting agency had to modify the original request by omitting D_9 and giving D_8 and D_{10} the cargo originally scheduled for D_9. In this manner, we go through each of the successive steps to the final solution.

Now, the generalized information flow model developed by Adams and Festa was based upon the idea that many systems have similar information flow requirements. Although the content of the data varies widely from system to system, the tasks involving the use of that data frequently recur both within a system and among systems.

As Hal has already mentioned, a point came, in the development of the general model, when it was necessary to examine closely the degree of fit between the model and some particular system task which was complete in itself (i. e., from initial data input to terminal decisions). As a result, our proposed simulation study of the logistics problem was selected for this examination. It satisfied the basic criterion mentioned above, and, furthermore, it was broken down into fairly small parts for analysis.

A basic conception in the development of the general model was that of the system verb, which, in each case, represented a particular type of action required of the system. The verbs were developed in a hierarchy of generality; thus, for example, in Appendix A, "list" sometimes appeared as a subheading under

"determine," which in turn was a part of "identify," while this latter was one aspect of the major function "assess."

The detailed comparison of the general model with the logistics model clearly illustrated two points. First, there was a considerable overlap between the two, especially at the more abstract level. Second, the attempt to conceive the particular parts of the logistics problem in terms of specific system verbs showed that many of these verbs recurred frequently, and that the conception of the problem in these terms led to a clearer, more easily programmed, breakdown of the problem parts, revealing, in fact, several places at which revision of the original problem flow diagram was required.

The flow diagram in Appendix A, entitled Generalized Information Flow Model: Transport Problem, illustrates the first point effectively. The numbers to the left of the chart refer to the step numbers of Figure 1 that are covered by the several system verbs. The shaded boxes indicate those system verbs that overlap the functions performed in the logistics problem of Figure 1; i.e., these areas of the general model that are in accord with the steps in the logistics problem.

Although the general model and the logistics model were independently conceived, the various parts of the latter can all be reasonably subsumed under the major system verbs, "data control," "assess," "decide," and "act." Not only this, but every one of the second-level verbs used under "assess" and "decide" (which are the core of the general model) occurs in some part of the specific logistics flow diagram. For example, "identify," the first of the second-level verbs under "assess," in the general model, includes the selection of cargo types and transport aircraft types appropriate for complying with the request. "Compare" involves the collection of relevant data on resource availability and destination capability. To "evaluate" is to use the data to ascertain if the request can be handled within the constraints of the situation. The major headings under "decide" are similarly represented, though in less detail, in the logistics flow diagram.

In addition, more than fifty lower-level verbs occur within the detailed flow diagram shown in Appendix A. These are, however, derived, with repetitions, from only twenty-three different verbs, with some of these essentially synonyms.

Of even more importance, the breakdown of the logistics model into specific system verbs permitted a detailed specification of the precise acts required, and showed clearly several discrepancies in the original logistics model. For example, the original showed the operator asking if changes in weather or intelligence data might make a solution possible at a time when the operator would often possess no data concerning weather or intelligence. The model was easily modified accordingly. A second instance of the operator's having insufficient information to ascertain the possibility of a solution was similarly handled by a simple change in the sequence of the diagram boxes. Finally, the original logistics model required a redundant assessment of solution possibility; this was simply deleted.

All of these errors could have been detected by a careful analysis of the logistics flow diagram. The point is simply that the logical discipline of the general model made the process of detecting these errors faster and easier, and may be expected to bear similar benefits in application to other systems or subsystems.

This model, as applied to the specific problem, thus reveals itself as a useful tool in the analysis of particular systems, and one whose power will be augmented by increasing use.

APPENDIX A

GENERALIZED INFORMATION FLOW MODEL: TRANSPORT PROBLEM

The following set of flow diagrams represents our first attempt to apply the concepts of the generalized decision-making model to a specific case; viz., the transport problem. Though limited in the degree of interaction between variables, it demonstrates the feasibility of the concept involved.

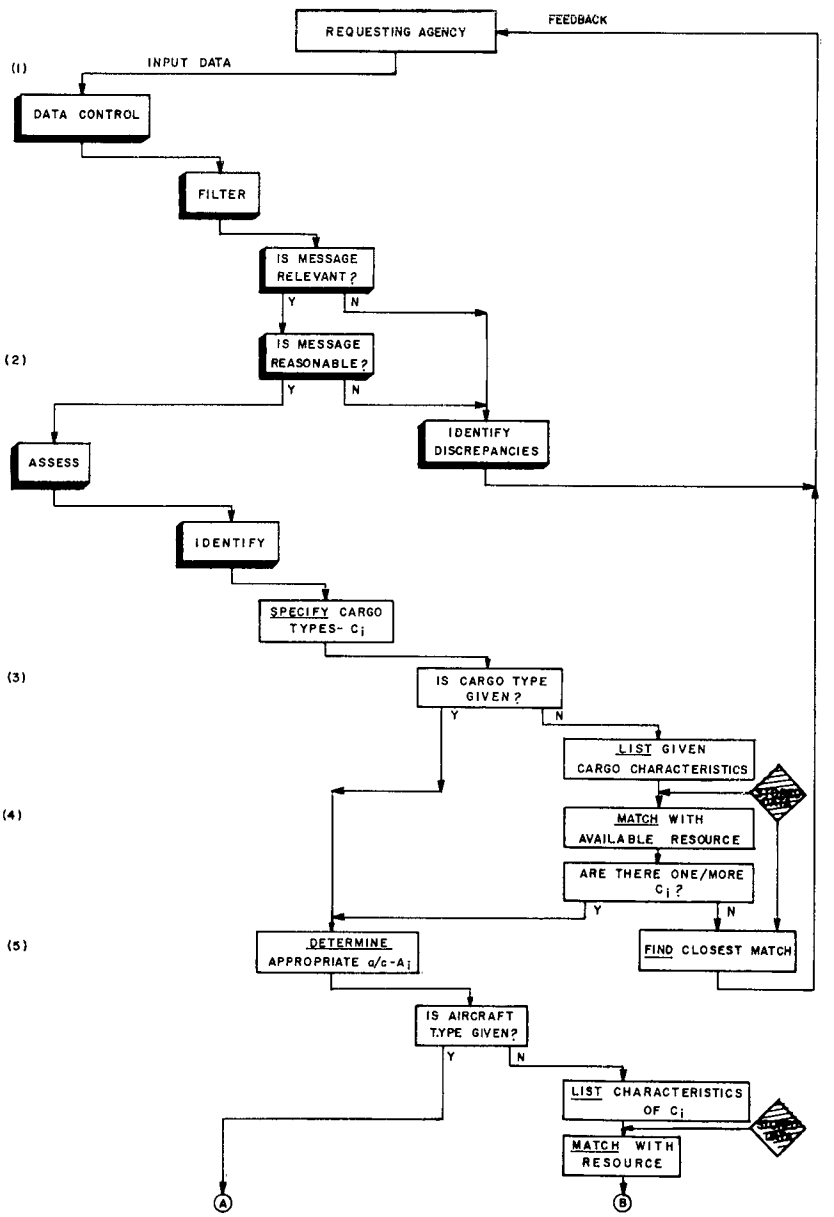
In this set of diagrams the extreme left-hand margin depicts the most abstract representation of functions performed in the system. The level of abstraction decreases from left to right until you reach the point where the flow diagram can be translated into a computer program (a point not reached in these diagrams).

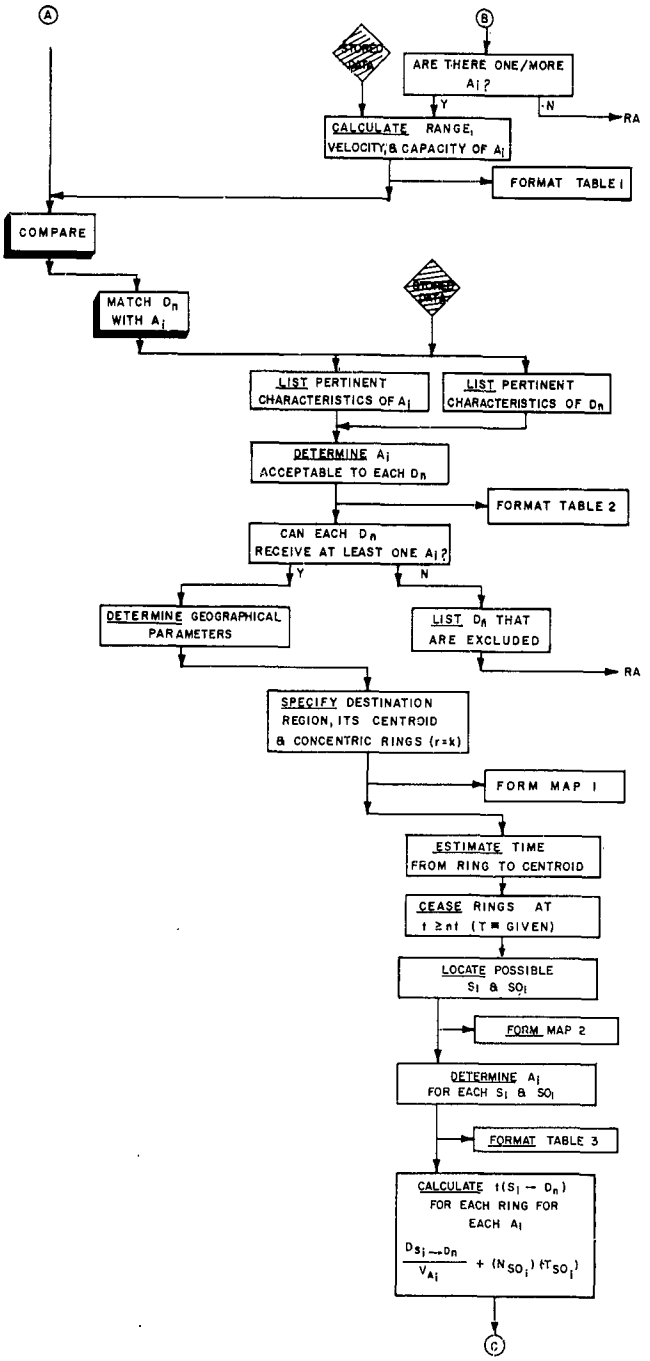
The black-outlined boxes represent those functions that are contained in both the general model and in the transport problem, i. e., the functional areas of overlap between the general and the specific.

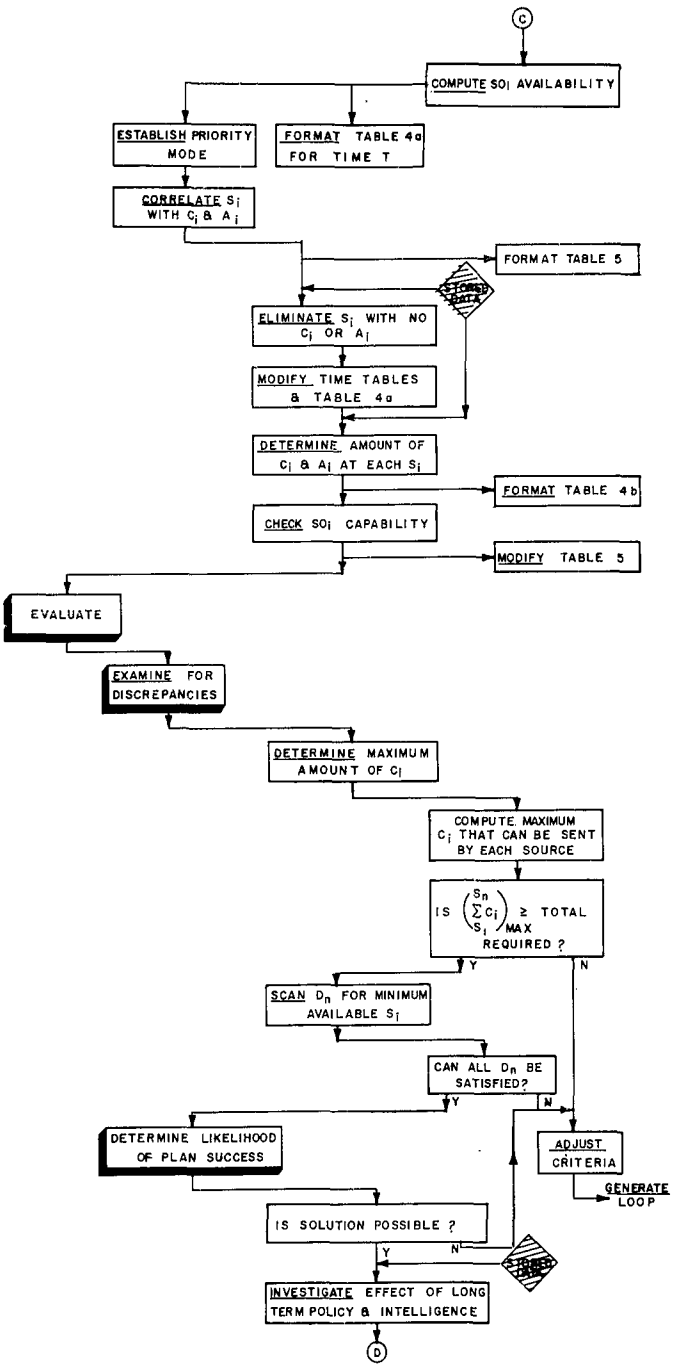
The numbers in parenthesis ( ) along the left-hand margin refer to the numbered boxes in the logistics flow diagram (Figure 1 in Section IV). These are given to indicate the transformation from the original logistics flow diagram to the framework of the general model.

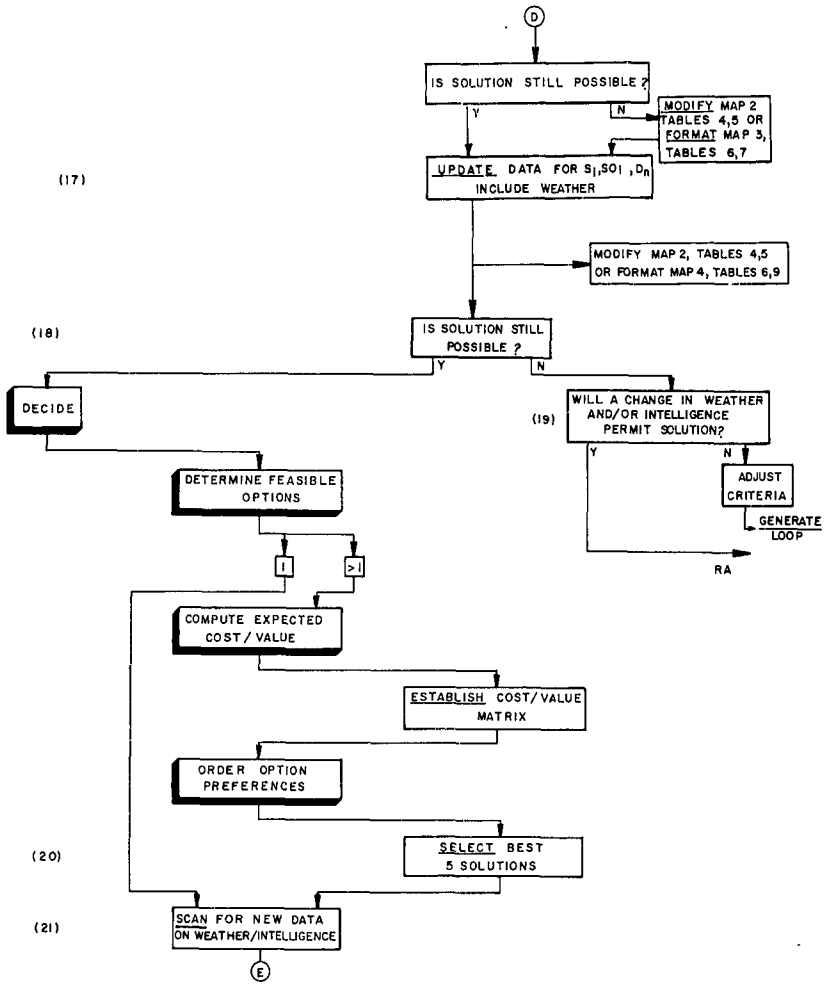
Within each box the initial verb is underlined to emphasize our contention that functions performed in systems can be described by a series of action verbs common to several systems rather than only one system.

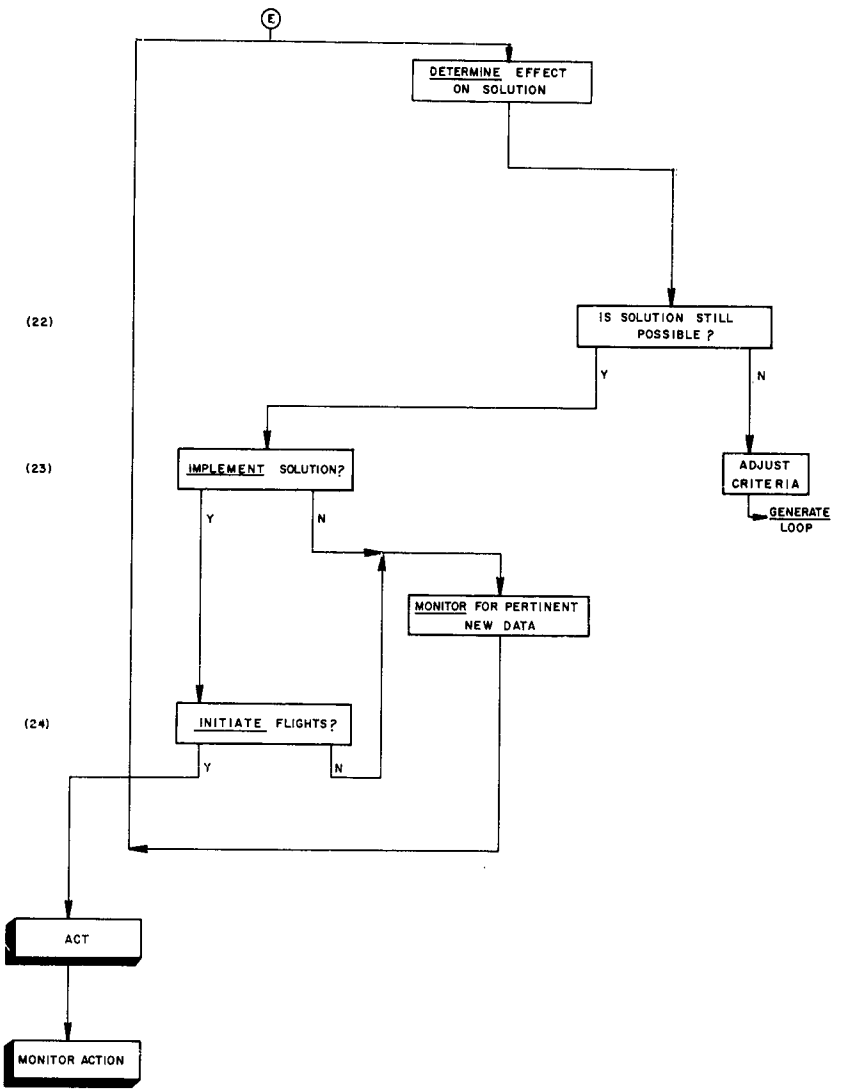
GENERALIZED INFORMATION FLOW MODEL:
TRANSPORT PROBLEM



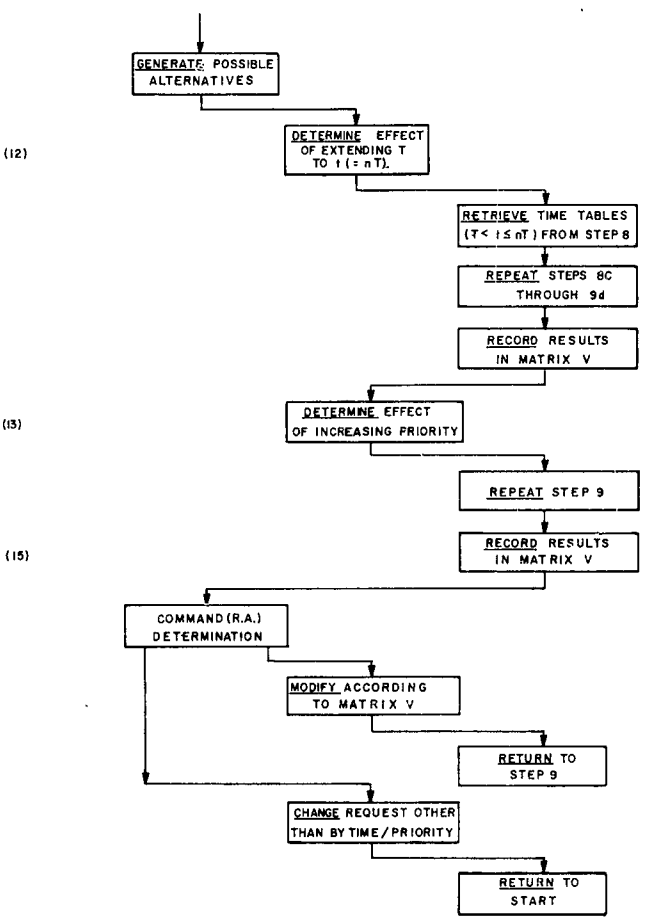

A
B
STORED DATA
ARE THERE ONE/MORE A_i?
Y
N
RA
CALCULATE RANGE, VELOCITY, & CAPACITY OF A_i
FORMAT TABLE 1
COMPARE
(6)
MATCH D_n WITH A_i
STORED DATA
LIST PERTINENT CHARACTERISTICS OF A_i
LIST PERTINENT CHARACTERISTICS OF D_n
DETERMINE A_i ACCEPTABLE TO EACH D_n
FORMAT TABLE 2
(7)
CAN EACH D_n RECEIVE AT LEAST ONE A_i?
Y
N
(8)
DETERMINE GEOGRAPHICAL PARAMETERS
LIST D_n THAT ARE EXCLUDED
RA
SPECIFY DESTINATION REGION, ITS CENTROID & CONCENTRIC RINGS (r=k)
FORM MAP 1
ESTIMATE TIME FROM RING TO CENTROID
CEASE RINGS AT $t \geq nt$ (T = GIVEN)
LOCATE POSSIBLE S_I & SO_i
FORM MAP 2
DETERMINE A_i FOR EACH S_I & SO_i
FORMAT TABLE 3
CALCULATE $t(S_I - D_n)$ FOR EACH RING FOR EACH A_i
$\frac{D_{S_i \rightarrow D_n}}{V_{A_i}} + (N_{SO_i})(T_{SO_i})$
C


C
COMPUTE SOI AVAILABILITY
(8c)
ESTABLISH PRIORITY MODE
FORMAT TABLE 4a FOR TIME T
(9)
CORRELATE S_i WITH C_i & A_i
FORMAT TABLE 5
STORED DATA
ELIMINATE S_i WITH NO C_i OR A_i
MODIFY TIME TABLES & TABLE 4a
DETERMINE AMOUNT OF C_i & A_i AT EACH S_i
FORMAT TABLE 4b
CHECK SO_i CAPABILITY
MODIFY TABLE 5
EVALUATE
EXAMINE FOR DISCREPANCIES
DETERMINE MAXIMUM AMOUNT OF C_i
COMPUTE MAXIMUM C_i THAT CAN BE SENT BY EACH SOURCE
IS $\left(\sum_{S_i}^{S_n} C_i\right)_{MAX} \geq$ TOTAL REQUIRED ?
Y
N
SCAN D_n FOR MINIMUM AVAILABLE S_i
CAN ALL D_n BE SATISFIED?
Y
N
DETERMINE LIKELIHOOD OF PLAN SUCCESS
ADJUST CRITERIA
GENERATE LOOP
(11)
IS SOLUTION POSSIBLE ?
Y
N
STORED DATA
(16)
INVESTIGATE EFFECT OF LONG TERM POLICY & INTELLIGENCE
D


D
IS SOLUTION STILL POSSIBLE ?
Y
N
MODIFY MAP 2 TABLES 4,5 OR FORMAT MAP 3, TABLES 6,7
(17)
UPDATE DATA FOR S_I,SO_I ,D_R INCLUDE WEATHER
MODIFY MAP 2, TABLES 4,5 OR FORMAT MAP 4, TABLES 6,9
(18)
IS SOLUTION STILL POSSIBLE ?
Y
N
DECIDE
(19)
WILL A CHANGE IN WEATHER AND/OR INTELLIGENCE PERMIT SOLUTION?
Y
N
ADJUST CRITERIA
GENERATE LOOP
RA
DETERMINE FEASIBLE OPTIONS
1
>1
COMPUTE EXPECTED COST / VALUE
ESTABLISH COST/VALUE MATRIX
ORDER OPTION PREFERENCES
(20)
SELECT BEST 5 SOLUTIONS
(21)
SCAN FOR NEW DATA ON WEATHER/INTELLIGENCE
E


E
DETERMINE EFFECT ON SOLUTION
(22)
IS SOLUTION STILL POSSIBLE ?
Y
N
(23)
IMPLEMENT SOLUTION?
ADJUST CRITERIA
GENERATE LOOP
Y
N
MONITOR FOR PERTINENT NEW DATA
(24)
INITIATE FLIGHTS?
Y
N
ACT
MONITOR ACTION

GENERATE LOOP



APPENDIX B

A DETAILED SAMPLE PROBLEM*

"Request 1. from the Requesting Agency, arriving at 0300Z 1 March 1961. Send 100 X-kiloton 1500-mile IRBMs to 10 specified destinations. Start at 0000Z 8 March 1961. End by 0000Z 11 March 1961. Priority level is 4.

D_1 to get 10	D_6 to get 5
D_2 to get 10	D_7 to get 15
D_3 to get 5	D_8 to get 15
D_4 to get 10	D_9 to get 5
D_5 to get 5	D_{10} to get 20."

In this request, the exact cargo is unspecified. Any missiles that fit the operational requirements can be used.

The cargo movement is not to begin for a week. This implies that there is time available for planning the job. The job itself must, however, be completed within 72 hours of its onset.

For priority level, a scale from 1 to 5 is tentatively assumed, such that level 1 is assigned to all resources (aircraft, cargo, fuel, facilities, personnel, etc.) that are for use in general war, while level 5 is assigned to resources that can be used on comparatively unessential jobs. To assign priority level 4 to this job, is to tell the CP that all resources earmarked for level 4 or 5 may be drawn upon to implement the task.

*To preserve generality throughout this problem, the codes C_i, A_i, D_i, SO_i are used instead of specific names to represent, respectively, the ith cargo, aircraft type, source base, destination base and stopover base.

The first question asked by the CP is, "Is this request feasible?" In this problem, the first answer given RA to this question is, "No. We know that during 8 and 9 March 1961 other tasks with higher priority will be occupying most of the available transport aircraft. Suggest that this task be done from 0000Z 10 March 1961 to 0000Z 13 March 1961." This step is essentially a human filtering process; a commander whose familiarity with the operation permits him to make this judgement. For example, the RA sends a reply arriving at 0700Z 1 March 1961, as follows:

"Request 2. In Request 1:

(1) Delete phrase 'Start at 0000Z 8 March 1961. End by 0000Z 11 March 1961.'

(2) Insert in place phrase 'Start at 0000Z 10 March 1961. End by 0000Z 13 March 1961.'

(3) Follow revised Request 1."

The CP, upon receiving Request 2, again asks Step 2, "Is this request feasible?" This time, the answer is, "Yes." Step 3 is, "Is cargo specified in the request?" The answer, "No," initiates Step 4, "What cargoes meet the operational requirements specified in the request?" To answer, the CP checks files of missile characteristics, and finds that it must tell RA, "None. There are, however, X-kiloton 1000-mile IRBMs, or 2X-kiloton 1500-mile IRBMs available." The RA then sends Request 3, which arrives at 1800Z 1 March 1961:

"Request 3. Supersedes Request 2.

Send 70 2X-kiloton 1500-mile IRBMs to 10 specified destinations. Start at 0000Z 10 March 1961. End by 0000Z 13 March 1961. Priority level is 4.

D_1 to get 7	D_6 to get 3
D_2 to get 7	D_7 to get 10
D_3 to get 3	D_8 to get 11
D_4 to get 7	D_9 to get 4
D_5 to get 4	D_{10} to get 14."

The CP once more asks Step 2, with the answer, "Yes." Step 3 is answered, "No"; then Step 4 is answered this time, "Cargoes C_1 and C_2 meet requirements." Step 5 asks, "What types of transport aircraft carry C_1 or C_2, and how many of each can be carried per aircraft?" The CP checks files for the answer here. These files may simply give relevant cargo and aircraft characteristics, such as cargo weight and linear dimensions, payload of the aircraft, and space available aboard. If this is the available data, capacities must be calculated. In this example, three types of aircraft are found to be able to carry units of C_1 or C_2, and the answer is:

"A_1 has 2000-mile range, can carry 5 C_1 or 7 C_2 or a mix.
A_2 has 1400-mile range, can carry 2 C_1 or 3 C_2 or a mix.
A_3 has 1000-mile range, can carry 1 C_2 only."

These data are set forth in Table 1.

Table 1

Range and Cargo Capacity of Relevant Aircraft Types

AIRCRAFT TYPE	RANGE (mi)	CARGO CAPACITY (UNITS)	
		C_1	C_2
A_1	2000	5	7
A_2	1400	2	3
A_3	1000	—	1

Step 6 asks which destination airfields are open to which types of aircraft; A_1, A_2, and A_3. The information is drawn from the files on airbase characteristics and aircraft requirements. It is learned that each destination can receive all three aircraft types with the following exceptions:

D_2 cannot receive A_1s.
D_6 cannot receive A_1s nor A_2s.
D_9 cannot receive any of these aircraft.

Because D_9 is closed to these aircraft, Step 7 means that the cargo cannot be moved there by any means under CP jurisdiction. The CP so informs the RA, which sends back Request 4 at 2200Z 1 March 1961:

"Request 4. In Request 3:

(1) Delete phrase 'D_8 to get 11, D_9 to get 4, D_{10} to get 14.'

(2) Insert in place phrase 'D_8 to get 13, D_{10} to get 16.'

(3) Follow revised Request 3."

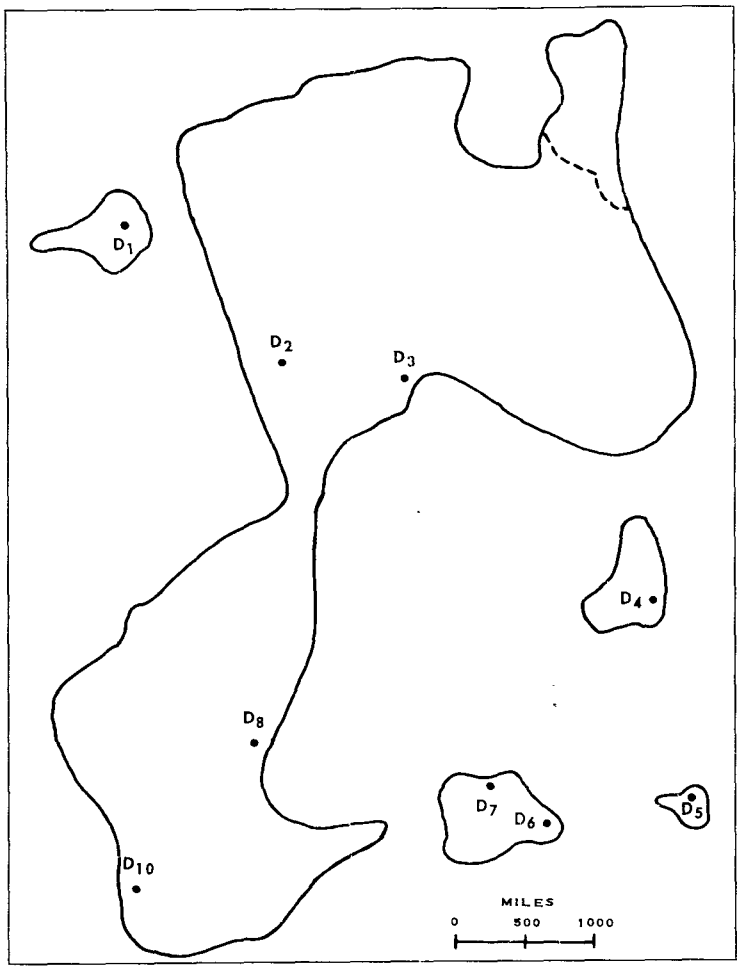
The CP then asks Steps 2 to 5 again with the same answers, and Step 6 is formulated with D_9 excluded. Map 1 and Table 2 are then prepared.

Map 1 covers an imaginary territory with the destinations marked on it. Table 2 shows which aircraft types can land at each destination. Step 7, this time, is answered affirmatively.

At this point, by asking Step 8, the CP estimates a region within which all sources, stopovers, and destinations to be used seem likely to be found. This is a judgment decision, not simply a following of rules. Map 1 may or may not include the entire region selected; in the present problem it does, but if it did not, a larger map would have to be prepared and used as a new Map 1.

Step 9 is a search for data on location and quantity of cargo and aircraft, location and characteristics of potential stopovers (including sources), a time estimate on the trip from each possible source to each destination using each aircraft type, and pertinent current information. The CP thus proceeds to Step 10, which involves the preparation of Map 2 and Tables 3, 4, and 5.

Map 2 shows as a source (S_i) each location which has at least one relevant cargo unit and one transport aircraft which can carry it, and can send the cargo

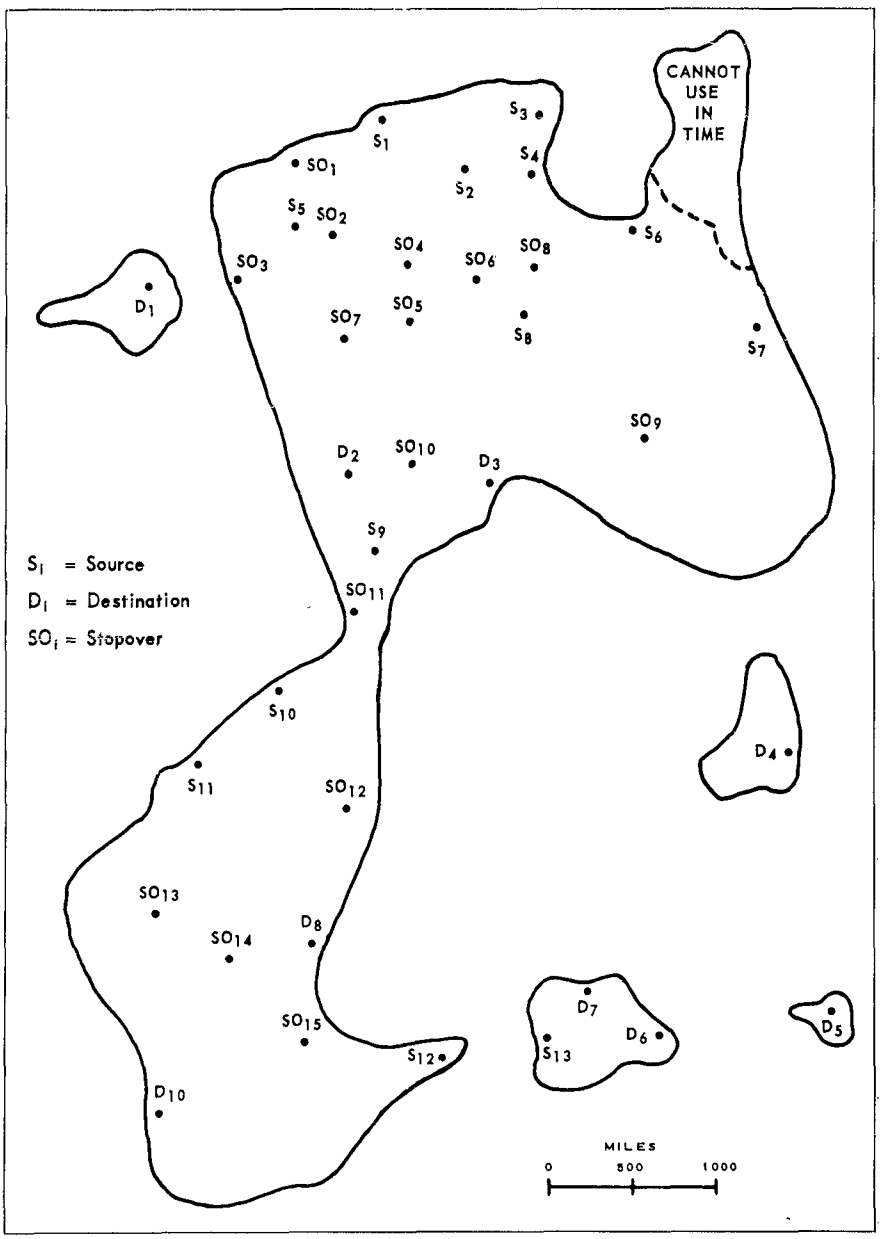


MAP I. DESTINATIONS

Table 2

Aircraft Types which can be Received at Each Destination

AIRBASE	A_1	A_2	A_3
D_1	x	x	x
D_2		x	x
D_3	x	x	x
D_4	x	x	x
D_5	x	x	x
D_6			x
D_7	x	x	x
D_8	x	x	x
D_{10}	x	x	x



MAP 2. SOURCES, DESTINATIONS, AND STOPOVERS

Table 3

Aircraft Types which can be Received at Each Airbase

AIRBASE	A_1	A_2	A_3
S_1	x	x	x
S_2	x	x	x
S_3	x	x	x
S_4	x	x	x
S_5	x	x	x
S_6	x	x	x
S_7	x	x	x
S_8		x	x
S_9		x	x
S_{10}	x	x	x
S_{11}	x	x	x
S_{12}	x	x	x
S_{13}	x	x	x
SO_1	x	x	x
SO_2	x	x	x
SO_3	x	x	x
SO_4	x	x	x
SO_5	x	x	x
SO_6	x	x	x
SO_7			x
SO_8	x	x	x
SO_9	x	x	x
SO_{10}		x	x
SO_{11}	x	x	x
SO_{12}		x	x
SO_{13}	x	x	x
SO_{14}	x	x	x
SO_{15}	x	x	x

Table 4

Transport Times for Aircraft Types and Resource Availability at Sources

SOURCES	DESTINATIONS BY AIRCRAFT TYPE																				CARGO UNITS AVAIL.		AIRCRAFT AVAIL.			MAX. CAPACITY	
	D_1			D_2			D_3			D_4			D_5			D_6			D_7			D_8			D_{10}									
	A_1	A_2	A_3	A_1	A_2	A_3	A_1	A_2	A_3	A_1	A_2	A_3	A_1	A_2	A_3	A_1	A_2	A_3	A_1	A_2	A_3	A_1	A_2	A_3	A_1	A_2	A_3	C_1	C_2	A_1	A_2	A_3	C_1	C_2
S_1	1	2	3	x	2	3	1	2	3	2	x	x	3	x	x	x	x	x	3	x	x	2	3	x	2	x	x	10	10	3	5	2	10	10
S_2	1	2	3	x	2	3	1	2	3	2	x	x	3	x	x	x	x	x	3	x	x	2	3	x	2	x	x	12	9	1	1	6	5	9
S_3	1	2	x	x	2	3	1	2	3	2	x	x	3	x	x	x	x	x	3	x	x	2	3	x	2	x	x	8	6	3	3	1	8	6
S_4	x	2	x	x	2	x	x	2	x	x	x	x	x	x	x	x	x	x	x	x	x	x	3	x	x	x	x	5	–	–	1	–	2	–
S_5	1	1	2	x	1	2	1	2	3	2	x	x	3	x	x	x	x	x	3	x	x	2	3	x	2	x	x	13	10	1	5	2	10	9
S_6	2	2	x	x	2	3	1	2	2	2	x	x	2	x	x	x	x	x	3	x	x	2	3	x	2	x	x	3	1	2	2	4	3	1
S_7	x	3	x	x	2	x	x	2	3	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	5	5	–	3	6	5	5
S_8	x	2	3	x	1	2	x	1	1	x	x	x	x	x	x	x	x	x	x	3	x	x	3	x	x	3	x	10	8	–	6	8	10	8
S_9	x	2	2	x	1	1	x	1	1	x	x	x	x	x	x	x	x	x	x	3	x	x	2	3	x	2	x	–	3	–	1	1	–	3
S_{10}	1	x	3	x	x	2	1	x	2	x	x	x	3	x	x	x	x	x	2	x	x	1	x	2	1	x	3	2	16	1	–	9	–	16
S_{11}	x	3	x	x	2	x	x	2	x	x	x	x	x	x	x	x	x	x	x	2	x	x	1	x	x	2	x	1	5	–	2	–	1	4
S_{12}	x	x	x	x	3	x	x	3	x	x	x	x	x	x	x	x	x	2	x	1	1	x	1	1	x	1	2	3	2	–	2	1	3	2
S_{13}	3	x	x	x	x	x	2	x	x	3	x	x	2	x	x	x	x	1	1	x	1	1	x	2	1	x	3	–	5	1	–	3	–	5
Cargo Units Required	7			7			3			7			4			3			10			13			16									
																											SUM:	72	80	12	31	43	57	78

For columns D_1 through D_{10} cell entries are times (in days) required for job (including loading, flying, unloading, refueling, etc.).
x = cannot be done (for whatever reason).

Table 5

Possible Routings from Each Source to Each Destination Using Each Aircraft Type

FROM	TO	A/C	SOME POSSIBLE STOPOVERS
S_1	D_1	A_1	---- (no stopover needed)
S_1	D_1	A_2	SO_1, SO_2, S_5, SO_3.
S_1	D_1	A_3	SO_1-SO_3, SO_2-SO_3, S_5-SO_3.
S_1	D_2	A_2	SO_2, SO_4, SO_5, SO_6.
S_1	D_2	A_3	SO_2-SO_7, SO_4-SO_7, SO_4-SO_5, SO_6-SO_5.
S_1	D_3	A_1	----
S_1	D_3	A_2	SO_4, SO_5, SO_6, SO_8.
S_1	D_3	A_3	SO_4-SO_5, SO_6-SO_8, S_2-SO_8-S_8.
S_1	D_4	A_1	S_2-SO_9, S_3-SO_9, S_4-SO_9, S_4-S_7, SO_6-SO_9, SO_8-SO_9.
S_1	D_5	A_1	S_2-SO_9-D_4, S_3-SO_9-D_4, S_4-SO_9-D_4, S_4-S_7-D_4, SO_6-SO_9-D_4, SO_8-SO_9-D_4.
S_1	D_7	A_1	Two Main Lines: 1) Incl. SO_9-D_4-D_5. 2) Incl. D_2-S_{10}, SO_5-SO_{11}-S_{11}.
S_1	D_8	A_1	incl. D_2-S_{10}, SO_5-SO_{11}-S_{11}.
S_1	D_8	A_2	SO_2-D_2-S_{10}, SO_4-D_2-S_{10}, SO_6-D_2-S_{10}, SO_5-S_9-S_{11}, SO_5-D_3-SO_{11}-SO_{12}.
S_1	D_{10}	A_1	Incl. D_2-S_{10}-SO_{13} (SO_{14}), SO_5-SO_{11}-S_{11}-D_8 (SO_{13}, SO_{14}).
S_2	D_1	A_1	
S_2	D_1	A_2	
.	.	.	
.	.	.	
.	.	.	etc.
S_{13}	D_{10}	A_1	
S_{13}	D_{10}	A_3	

Code:
1. Single entries (SO_1, SO_2) refer to single stops within those routes from S_i to D_i which require just one stop.
2. Plural entries (SO_1-SO_3, SO_2-SO_3) refer to series of stops within those routes which require more than one stop.
3. Parenthetic entries (SO_{13}(SO_{14})) refer to alternate stops.

to some destination specified within the time limit. A stopover (SO_i) is an airbase that is neither a source nor a destination but that can receive and send off at least one of the relevant aircraft types, and could be used as a part of some source-to-source destination route within the time limit. Aircraft may also stop at sources or destinations if these are equipped to handle the aircraft. (As mentioned above, the sample problem assumes that no aircraft will load cargo except at its starting point, nor unload except at its final destination.)

Table 3, an extension of Table 2, shows which of the selected aircraft types can land at each source, stopover, and destination in the region of interest.

The main portion of Table 4 shows a (Source) x (Destination) matrix, with each cell including an entry for each aircraft type. This entry is "x" if:

(a) either the source or the destination cannot receive the aircraft type;
(b) required stopovers are not available for the trip;
(c) the trip requires longer than the time limit; or
(d) the source has none of the aircraft type on hand.

Otherwise the entry is a digit equal to the days needed for the trip. The bottom row gives the cargo requirements for each destination.

The right side of the table shows:

(a) the amount of each cargo available at each source (under column heading CARGO UNITS AVAIL.):
(b) the number of each aircraft type available at each source (under column heading AIRCRAFT AVAIL.); and
(c) the maximum of available cargo and available aircraft capacity (under column heading MAX CAPACITY).

This last depends on the capacity data from Table 1.

Table 5 shows the various possible routings from each source to each destination using each aircraft type. In this table, for example, a notation such as:

FROM:	TO:	A/C	SOME POSSIBLE ROUTES
S_1	D_1	A_1	- - - - - - - - - - -
S_1	D_1	A_2	SO_1, SO_2, S_5, SO_3
S_1	D_1	A_3	SO_1-SO_3, SO_2-SO_3, S_5-SO_3

indicates that to ship from source 1 (S_1) to destination 1 (D_1) via aircraft type 1 (A_1), no stopovers are needed and the great circle route is implied. To do the same with A_2, any of four single stopovers will do; namely, stopover 1, 2, 3, or source 5. To ship via A_3, each routing must use two stopovers: first, stopover 1 or 2, or source 5; and second, in every case, stopover 3.

In this problem, the CP learns that only 29 C_1 and 12 C_2 are available at priority level 4 or lower and can be moved with the three-day time limit imposed. Since 70 C_i are needed, no solution is possible to request 4, and the CP enters Steps 11 to 15, essentially asking, "Will releasing some constraint permit a solution?"

Step 12 asks whether a release in the time constraint would lead to adequate resources to settle the problem. In the present example, the CP learns that if four days are allowed instead of three, 41 C_1 and 37 C_2 units will be available at priority level 4 or lower. The data are recorded, and Step 13 is entered, asking about a release in priority constraint. The answer here is that, if priority level 3 is included, 57 C_1 and 78 C_2 can be moved within three days.

Since a change of one grade in time or of one in priority leads to a possible solution, Step 14, concerned with joint variation of time and priority, is tested only for trade-off (tightening one constraint while relaxing the other). In this problem, no trade-off leads to a possible solution.

The RA is thus offered the choice of raising the priority level, increasing the time limit, or reducing the quantity of cargo required. The first of these is

chosen, and Request 5 comes in at 1100Z 2 March 1961 as follows:

"Request 5. In Request 3, revised by Request 4:

(1) Delete phrase 'Priority level is 4.'

(2) Insert in place phrase 'Priority level is 3.'

(3) Follow revised Request 3."

The CP then finds the answers to Steps 2 and 9 unchanged, while Steps 10 and 11 now have the outcome, "Solution possible."

The CP, proceeding to Step 16, ascertains whether long-term policy and intelligence data (at least two weeks old) restrict the field of possible routes and therefore the problem solutions. In the present example, there are overflight restrictions that affect the situation. The region of restriction is plotted in Map 3 (similar to Map 2). The routes still possible are set forth in Tables 6 and 7 (similar to Tables 4 and 5, respectively).

The solution to the routing problem consists of the selection of the best possible resource allocation to move the cargoes. Step 17 starts the CP on this trail. In that step, the CP obtains detailed current information about the resource and other status of all relevant airbases. (Current data are over six hours but less than two weeks old.) This permits the generation of Map 4 (similar to Map 2) and Tables 8 and 9 (similar to Tables 4 and 5, respectively), which embody any further reductions imposed by the new information. Step 18 then consists in finding whether no solutions, one solution, or several solutions are available within the new constraints. Table 10, whose cell entries are dollar cost rather than time required for each trip, is otherwise similar to Table 4, and is used to derive and order these possible solutions. In the present example, with its large surplus of cargoes and aircraft available, several solutions are possible. In Step 20, the CP puts the best five solutions in order, in terms of the assumed basic criterion of dollar cost of implementation. (There is nothing sacred about this criterion; others will serve equally well.)

The very best one of these five solutions is selected. In Step 21 the momentary (less than six hours old) data on weather and intelligence are brought in, and their effects recorded on Map 5 (similar to Map 2) and Tables 11, 12, and 13 (similar to Tables 4, 5, and 10, respectively), which may show further changes from the last level plotted. Step 22 asks whether the best solution has been changed in any way by the momentary data; in the present problem it has not. Since this is true, the CP advances to Step 23, implementation of the solution, and sends messages to operators in charge, giving instructions for carrying out the job. At Step 27, the question whether cargo loading and flights are to be initiated is settled, "No" (it is too soon, nearly a week early), and a return to Step 21 occurs. If there are momentary data, they are recorded on a new or revised Map 5, and Tables 11, 12, and 13; and the old ones, if useless, are discarded. The CP again advances through Steps 22 and 23 to ask, again, whether flight should be initiated. As soon as "Yes" becomes the answer, the routing phase ends, and plan monitoring, not here examined, begins.

APPENDIX C

CONTRIBUTORS

Harold W. Adams was born in Bronxville, New York on March 13, 1925. He received his B. S. degree in 1949 from the University of Connecticut, his M. A. degree in 1950 from the University of North Carolina, and his Ph. D. degree in History and Political Science in 1954 from Clark University. From 1943 to 1946 he served in the U. S. Navy. From 1951 to 1952 he was a Teaching Fellow at Clark University. He was on the staff of the Director of Electronics Research, Air Force Cambridge Research Center in Bedford, Massachusetts from 1952 to 1955. From 1955 to 1958 he was Personnel Manager and Administrator for Hycon Eastern, Inc. He joined MITRE Corporation in 1958 as Director of Personnel. In 1959 he became a Staff Member of the System Sciences Department. He is presently Subdepartment Head of Design Methodology.

Geoffrey Gordon was born in London, England in May 1924. He received his B. Sc. degree in Physics in 1946 and his M. A. degree in Mathematics in 1949 from London University, England. From 1950 to 1955 he was a member of the Research Staff at Research Laboratories of the General Electric Co., England, on the study of missile guidance and control systems. On coming to the United States in 1955 he was at Westinghouse till 1956. From 1956 to 1960 he was a Staff Member of the Bell Telephone Laboratories engaged in the simulation study of switching systems and communications networks. He joined the IBM Corporation in 1960 as a Staff Member in the Advanced Systems Development Division in White Plains, New York. He is currently Manager of Simulation Development responsible for developing simulation programs required to support studies in new systems concepts. He is a member of the Association of Computer Machinery.

Charles S. Morrill was born in Boston, Massachusetts on September 17, 1927. In 1949 he received his B. S. degree in Experimental Psychology from Tufts University. From 1949 to 1950 he attended the Sorbonne. In 1951 he received his M. A. from Columbia University in Tests and Measurements, Psychology. He was a Research and Academic Assistant from 1950 to 1952 at Columbia University. From 1952 to 1953 he was a member of the U. S. Air Force. He served as a member of the Staff of Tufts University, Department of the Systems Analysis, from 1953 to 1954. From 1954 to 1957 he was a consultant for Globe, Byron and Acorn Corporation. In 1957 he became a Staff Member of Radio Corporation of America, Missile Electronics and Controls Department. He joined the Staff of MITRE Corporation in 1959 as a member of the Human Factors Subdepartment of the System Sciences Department. Mr. Morrill is a member of the American Psychological Association, International Ergonomics, Kappa Delta Pi, Phi Delta Kappa and Psi Chi.

Peter H. Ten Eyck was born in London, England on March 30, 1932. He received his A. B. degree in Physics in 1953 from Princeton University. His graduate work was in Social Psychology and he received his Ph. D. from Boston University in 1960. He served with the U. S. Army from 1953 to 1955 as a Statistician. From 1956 to 1960 he was associated with Boston University in research in physics and psychology. He joined the MITRE Corporation in 1960 as a Staff Member of the Human Factors Subdepartment of the System Sciences Department. Dr. Ten Eyck is a member of the American Psychological Association and Psychometric Society.

Irving A. Wallach was born in New York, New York on December 3, 1919. He received his B. A. degree from New York University in 1947. From 1947 to 1950 he was a member of the Ph. D. program in Cultural Anthropology at Columbia University. He was a Social Science Analyst at the U. S. Information Agency in Washington, D. C. from 1951 to 1953. In 1953 he was a consultant for the Society for Applied Anthropology in New York. He served as a Research Staff Associate for Universal Pictures, Inc., in New York from 1953 to 1954. From 1955 to 1956 he was a Research Fellow at the Ford Foundation in New York. From 1957 to 1959 he was a Research Scientist

at the Special Operations Research Office in Washington, D. C. In 1959 he joined the System Development Corporation as a Human Factors Scientist in the SACCS Department in Paramus, New Jersey. He is presently Assistant to the Technical Director for technical product control. Mr. Wallach is a member of the Society for Applied Anthropology and the American Anthropological Association.